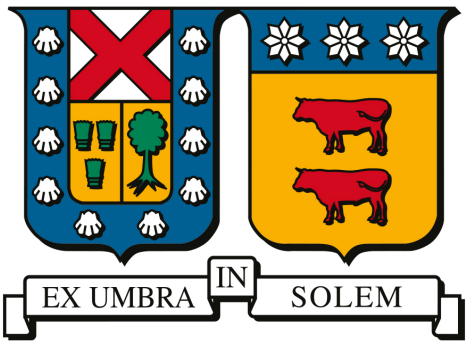
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO, CHILE



"Aplicando pruebas de software en contexto educacional: Caso ÚneteApp"

IAN LUCAS COOPER ABARZÚA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Pablo Cruz Navea
Profesor Correferente: Raúl Monge Anwandter

Octubre - 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Aplicando pruebas de software en contexto educacional: Caso ÚneteApp

Nombre del candidato(a): Ian Lucas Cooper Abarzúa

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central Valparaíso **Departamento:** Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Pablo Sebastián Cruz Navea, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

_____

_____

_____

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29-10-2025 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 29-10-2025 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas que me acompañaron a lo largo de mi trayectoria universitaria, haciendo de esta etapa un periodo tan entretenido como importante.

En primer lugar, a mi familia. A mi mamá, por su apoyo constante e incondicional, que fue fundamental para que pudiera enfocarme por completo en mis estudios. A mis hermanos, por estar siempre ahí, dándome ánimo y compartiendo momentos que me liberaban del estrés de la universidad. Y por supuesto, a mis mascotas, que siempre estuvieron entregándome su cariño.

También extiendo mi gratitud a las personas que conocí durante la carrera. A mis compañeros y amigos de la universidad, con quienes fue muy entretenido asistir a clases, compartir en los recesos y divertirnos, motivándonos mutuamente para seguir mejorando y aprendiendo. Un agradecimiento especial a mi pareja, a quien conocí en mi cuarto año de carrera, por su apoyo incondicional en todo lo relacionado con la universidad.

Adicionalmente, quiero agradecer a los profesores que me guiaron y apoyaron durante mí formación académica. En particular, a mi profesor guía, por su paciencia, orientación y valiosos consejos que fueron cruciales para la realización de este trabajo (y las conversaciones casuales que siempre eran entretenidas de escuchar).

Finalmente, una mención especial a mis amigos fuera de la universidad. Siempre es un gusto pasar un buen rato y disfrutar de su compañía, ya sea en persona (en partidos de fútbol o juntas) o en Discord para jugar un rato. Su amistad y cariño fueron vitales, sobre todo durante los primeros dos años de la carrera, que se realizaron en plena pandemia. Gracias por hacer esos momentos más llevaderos.

Resumen

El desarrollo de software en contextos educacionales es una práctica cada vez más frecuente, ya que permite acercar a los estudiantes a la práctica profesional de la ingeniería de software. Sin embargo, las restricciones de tiempo y recursos inherentes a estos entornos a menudo conducen a que tanto estudiantes como profesores prioricen requisitos funcionales por sobre aspectos vinculados a la calidad del producto, cuya evaluación puede fortalecerse mediante la aplicación de pruebas de software. Esta omisión en la aplicación de procesos de prueba formales compromete la calidad y robustez del software desarrollado.

En este trabajo de memoria se aborda esta problemática mediante el diseño, aplicación y evaluación de un proceso de pruebas de software en "ÚneteApp", una aplicación móvil desarrollada en el entorno educacional de un taller que culmina con la "Feria de Software" de la Universidad Técnica Federico Santa María. El proceso de pruebas se basa en el estándar ISO 29119:2021, y su aplicación se enfoca en analizar las implicancias de introducir prácticas de testing formales en un proyecto académico con limitaciones de desarrollo.

El análisis retrospectivo del proceso permitió no solo validar la funcionalidad de la aplicación, sino también identificar y analizar los hechos esperados y hechos no previstos que surgen al aplicar un proceso de pruebas formal en un contexto educacional. Los resultados ofrecen una visión sobre los desafíos y beneficios de esta práctica, y también se entregan recomendaciones para mejorar la calidad del software en iniciativas de software desarrolladas en contextos educacionales similares.

Palabras Clave – Pruebas de software, ISO 29119, Contexto educacional, Feria de Software, Calidad de software, Software de salud.

Abstract

Software development in educational contexts is an increasingly frequent practice, as it allows students to approach the professional practice of software engineering. However, the time and resource constraints inherent to these environments often lead both students and professors to prioritize functional requirements over aspects related to product quality, whose evaluation can be strengthened through the application of software testing. This omission in the application of formal testing processes compromises the quality and robustness of the developed software.

This graduation report addresses this issue through the design, application, and evaluation of a software testing process in "ÚneteApp", a mobile application developed in the educational environment of the "Software Fair" at Universidad Técnica Federico Santa María. The testing process is based on the ISO 29119:2021 standard, and its application focuses on analyzing the implications of introducing formal testing practices into an academic project with development limitations.

The retrospective analysis of the process allowed to validate the application's functionality and to identify and analyze the expected and unexpected events that arise when applying a formal testing process in an educational context. The results offer insights into the challenges and benefits of this practice, and recommendations are also provided to improve software quality in similar software initiatives developed in educational contexts.

Keywords – Software testing, ISO 29119, Educational context, Software Fair, Software quality, Health software.

Índice general

Índice de Figuras

Índice de tablas

Glosario

IEEE: *Institute of Electrical and Electronics Engineers,* es una organización profesional técnica dedicada a la estandarización y desarrollo de tecnologías.

SBP: *Sistema Bajo Prueba,* es el sistema bajo prueba.

SWEBOK: *Software Engineering Body of Knowledge,* es un documento que define el conocimiento esencial en ingeniería de software.

ISO: *International Organization for Standardization,* es una organización internacional que desarrolla y publica estándares internacionales.

ISTQB: *International Software Testing Qualifications Board,* es una organización que certifica profesionales en pruebas de software y promueve buenas prácticas en la industria.

UTFSM: *Universidad Técnica Federico Santa María,* es una universidad chilena reconocida por su enfoque en ingeniería y tecnología.

OMG: *Object Managment Group,* es una organización internacional que desarrolla estándares para la industria del software.

Capítulo 1

Introducción

1.1. Contexto y motivación

Las pruebas de software son una etapa crucial en el ciclo de vida del desarrollo de software, ya que permiten verificar que el sistema cumple con los requerimientos y especificaciones definidas. En un contexto educacional, como lo es el taller de desarrollo que culmina con la Feria de Software de la Universidad Técnica Federico Santa María (UTFSM), existen desafíos que dificultan esta tarea, además los proyectos de software desarrollados en esta instancia por los estudiantes suelen enfocarse en la funcionalidad de un prototipo final, dejando de lado aspectos importantes en términos de la calidad del producto.

1.2. Definición del problema

El problema que se aborda en esta memoria es el diseño, análisis y ejecución de pruebas de software para un producto de software desarrollado en un contexto educacional (software desarrollado como parte de las tareas asignadas en cursos de pregrado y postgrado), específicamente, proyectos de software desarrollados en el marco de la Feria de Software. A pesar de que los estudiantes adquieren conocimientos teóricos sobre pruebas de software, su aplicación práctica es limitada debido a diversas razones, como la falta de tiempo, recursos y la ausencia de un cliente activo durante el desarrollo del proyecto. En este contexto, se busca aplicar un proceso formal de pruebas de software, lo que implica el diseño, implementación y ejecución de casos de prueba orientados a mejorar la calidad del software desarrollado en el marco de la Feria de Software, específicamente en la aplicación ÚneteApp.

1.2.1. Facilitando las pruebas de software desarollado en un contexto educacional

ÚneteApp es una aplicación móvil desarrollada en un contexto educacional bajo el marco de la Feria de Software de la UTFSM. Debido a este entorno de desarrollo, la aplicación no cuenta con un proceso formal de pruebas de software. En este sentido, la propuesta consiste en aplicar un proceso formal, lo que implica el diseño, implementación y ejecución de casos de prueba con el propósito de mejorar la calidad del software desarrollado en la Feria de Software, particularmente en el caso de la aplicación ÚneteApp.

Asimismo, se busca realizar un análisis retrospectivo de los hechos esperados y hechos no previstos que emergen al aplicar un proceso de pruebas de software en un contexto educacional.

1.2.2. Objetivos

Objetivo general

Diseño, aplicación y evaluación de pruebas de un sistema de software desarrollado en un contexto educacional.

Objetivos específicos

- Definir y diseñar casos de prueba específicos para la aplicación ÚneteApp
- Aplicar pruebas de software para evaluar a la aplicación ÚneteApp
- Analizar hechos esperados y preconcebidos que emergen al aplicar pruebas de software a la aplicación ÚneteApp
- Analizar hechos observados no evidentes que emergen al aplicar pruebas de software

1.2.3. Estrategia de trabajo

Para cumplir con los objetivos planteados en esta memoria, se presenta la estrategia metodológica que guiará su desarrollo. En primer lugar, se levantará la arquitectura de ÚneteApp, basandose en lo implementado en Taller de Feria de Software poniendo énfasis en las historias de usuario y criterios de aceptación que fueron implementados. Posteriormente, se seguirá el proceso sugerido por la ISO/IEC 29119-2:2021 acerca del proceso de pruebas de software, específicamente la parte de "Dynamic Test Processes"1.1. Este proceso cuenta con las siguientes 4 etapas:

1. Proceso de diseño e implementación
2. Proceso de manejo de datos y ambiente de pruebas
3. Proceso de ejecución
4. Proceso de reporte de incidentes de pruebas

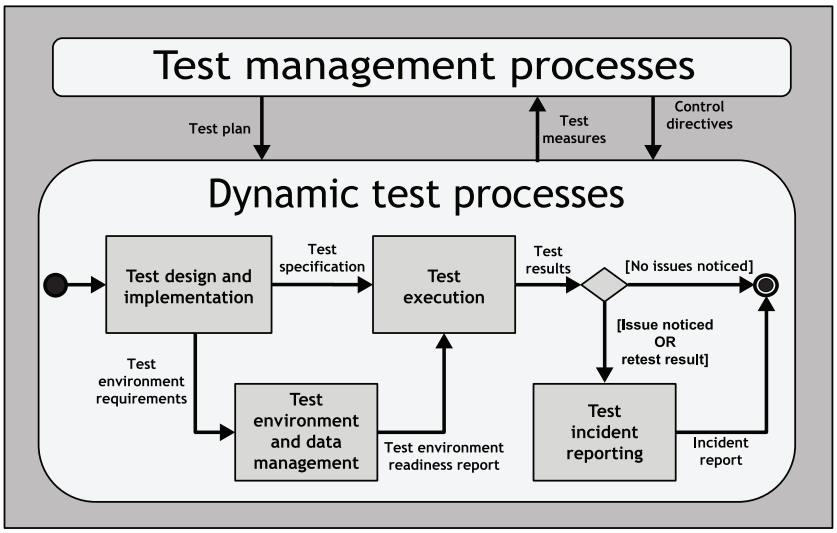


Figura 1.1: *Dynamic Test Processes* según ISO/IEC 29119-2:2021

Fuente: Estándar ISO/IEC 29119-2:2021 [18]

Luego de realizar el levantamiento de la arquitectura del sistema y terminar el proceso de pruebas, se realizará un análisis retrospectivo de todo el trabajo realizado, con el fin de identificar tanto los hechos esperados como los no previstos que emergen al aplicar pruebas de software en un contexto educacional. Este análisis permitirá reflexionar sobre la efectividad del proceso de pruebas y su impacto en la calidad del software desarrollado.

Para poder lograr satisfactoriamente los objetivos planteados, se utilizará el roadmap de aprendizaje de pruebas de software presentado en la figura 1.2, el cual guiará el proceso de aprendizaje y aplicación de pruebas de software a lo largo del desarrollo de la memoria.

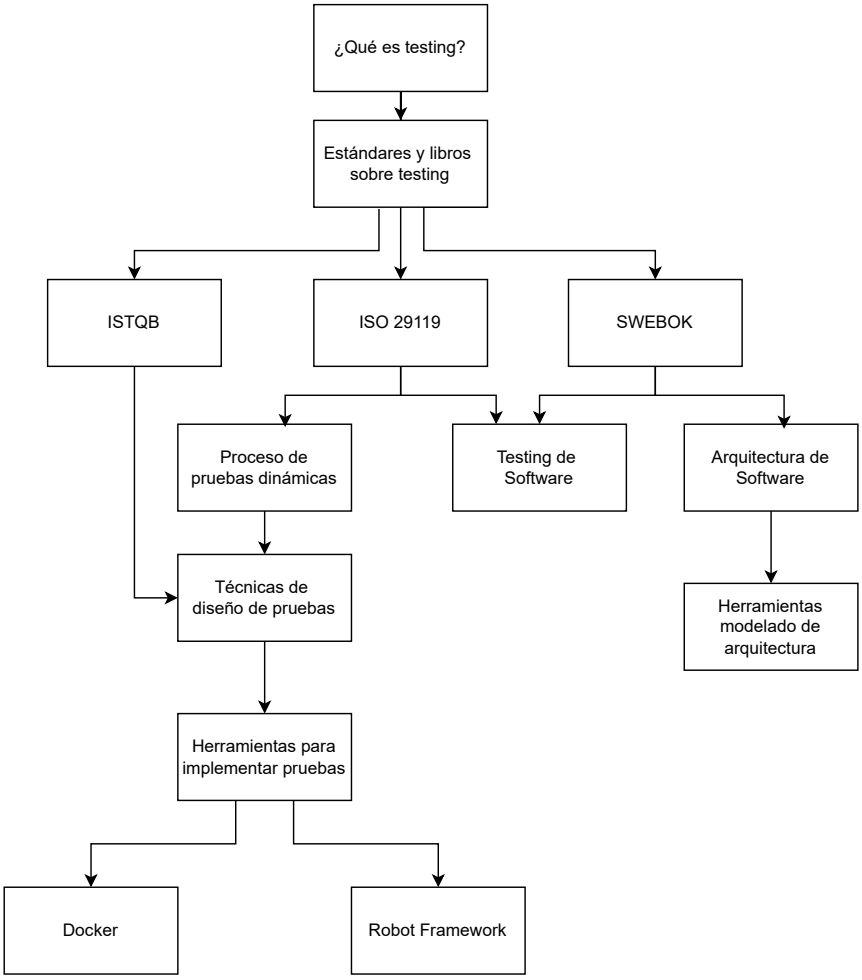


Figura 1.2: Roadmap de aprendizaje a aplicar en la memoria

1.2.4. Estructura del documento

El trabajo se estructura de la siguiente manera:

- Capítulo 1: se presenta el contexto y motivación, así como la definición del problema que aborda ÚneteApp, los objetivos planteados y la estrategia de trabajo.
- Capítulo 2: se describe en detalle el contexto de la Feria de Software de la UTFSM, se aborda la problemática que resuelve ÚneteAPP.
- Capítulo 3: se describe el marco conceptual y el estado del arte relacionado con las pruebas de software, arquitectura de software en contextos educacionales.
- Capítulo 4: se presenta el proceso de pruebas de software aplicado a la aplicación ÚneteApp, incluyendo el levantamiento de la arquitectura del sistema y el diseño e implementación de los casos de

prueba.

- Capítulo 5: se realiza un análisis retrospectivo de los hechos observados durante la aplicación del proceso de pruebas, identificando tanto los hechos esperados como los no previstos. También se mencionan dificultades encontradas y recomendaciones para desarrollos en contextos similares.
- Capítulo 6: se presentan las conclusiones del trabajo, así como las recomendaciones futuras asociadas a ÚneteApp.

Capítulo 2

Definición del problema

2.1. Contexto

La Feria de Software de la Universidad Técnica Federico Santa María es un hito anual del Departamento de Informática en el que equipos exhiben prototipos de software desarrollados durante el año académico. Desde el punto de vista formativo, constituye una instancia integradora que articula cursos y proyectos, promueve la innovación y el emprendimiento, y fortalece la vinculación con el medio mediante la interacción con visitantes y auspiciadores de la industria.

En este contexto se desarrolló *ÚneteApp*, una aplicación orientada a abordar dos problemáticas sociales específicas: (1) la **soledad y el aislamiento social**, especialmente en jóvenes, y (2) la **congestión en los sistemas de salud** a nivel nacional. La aplicación facilita la conexión entre personas con intereses comunes, promoviendo la interacción social y el bienestar emocional, y contribuye a descongestionar los servicios de salud mediante la implementación de la prescripción social.

2.2. Feria de Software en la UTFSM

La Feria de Software es una instancia clave en la formación de los estudiantes de Ingeniería Civil en Informática de la UTFSM, donde se busca aplicar los conocimientos adquiridos a lo largo de la carrera en proyectos reales que solucionan un problema del mundo real. A pesar de que la feria de software busca ser un primer acercamiento a la industria, existen dificultades inherentes a la naturaleza académica durante el transcurso de la actividad.

La Feria de Software se articula en dos asignaturas: **Gestión de Proyectos de Informática (INF-360)** [31], impartida en el noveno semestre, y **Taller de Desarrollo de Proyectos de Informática (INF-228)** [32], impartida en el décimo semestre. El proceso culmina con la presentación pública en la Feria, donde los equipos defienden su propuesta ante estudiantes, profesores, expertos del área y potenciales clientes, mostrando el prototipo final desarrollado.

2.2.1. Gestión de Proyectos de Informática (INF-360)

Descripción General de la Asignatura

La asignatura INF-360, se enfoca en la gestión de proyectos informáticos, orientada específicamente a la formulación y gestión de iniciativas de emprendimiento innovador basadas en Tecnologías de la Información (TI). Los estudiantes, a través de un trabajo en equipo, se sumergen en el análisis de procesos, métodos, técnicas y herramientas utilizadas en la formulación y gestión de proyectos, con un enfoque particular en contextos de licitaciones.

El objetivo principal es que los estudiantes formulen un proyecto que integre las etapas de inicio y planificación, basándose en una problemática real, lo que servirá de base para la continuación de dicho proyecto en un semestre posterior.

Metodología y Contenidos

La metodología de enseñanza se centra en el aprendizaje basado en proyectos. Los estudiantes participan en una "Licitación de Emprendimiento Innovador", donde planifican y comienzan a desarrollar un proyecto hasta alcanzar un prototipo inicial o PMV. Este proceso se complementa con la aplicación de conceptos vistos en clase, incluyendo aspectos éticos, legales y de negocios. Los contenidos temáticos cubren una amplia gama de conocimientos esenciales para la gestión de proyectos, tales como:

- Detección de oportunidades de negocios: Enfoque en el emprendimiento innovador con base en TI y el uso de la metodología Design Thinking.
- Contexto ético-legal: Fundamentos éticos, aspectos clave de la propiedad intelectual y contratos informáticos.
- Gestión de proyectos: El ciclo de vida de la gestión de proyectos, metodologías ágiles y el PMBOK, y técnicas de planificación que incluyen la gestión de alcance, tiempo y comunicaciones.
- Evaluación económica: Estructura de costos, fuentes de ingresos y evaluación financiera de proyectos informáticos.

Resultados de Aprendizaje y Contribución al Perfil de Egreso

La asignatura busca que los estudiantes logren una serie de resultados de aprendizaje, como formular propuestas de proyectos informáticos en contextos de licitación competitiva, analizar el marco ético y legal, aplicar técnicas de gestión basadas en estándares internacionales, y elaborar planes de proyecto detallados. Además, se espera que puedan evaluar económicamente las propuestas y construir prototipos iniciales.

En cuanto a su contribución al perfil de egreso, la asignatura INF-360 fomenta competencias específicas, como la capacidad de formular, desarrollar y gestionar proyectos informáticos orientados a la innovación y el emprendimiento. También fortalece competencias transversales, incluyendo la comunicación eficaz, la dirección de equipos de trabajo, la autonomía, el pensamiento crítico, la actualización profesional continua, el aseguramiento de la calidad y el fomento de una conducta ética y responsable.

Consideraciones académicas y contextuales (INF-360 — 9.º semestre)

Al momento de inscribir y cursar **INF-360**, conviene atender a cinco ámbitos: (1) *formación previa y prerrequisitos*, (2) *estructura y evaluación de la asignatura*, (3) *organización del proyecto y artefactos*, (4) *licitación y evaluación externa*, y (5) *carga académica y calendario*.

1. **Formación previa y prerrequisitos**
   - **Cobertura en pruebas de software:** no es obligatorio haber cursado o estar cursando alguna asignatura enfocada en *pruebas de software*; el nivel de conocimientos en *testing* puede ser distinto entre estudiantes.
2. **Estructura y evaluación de la asignatura**
   - **Evaluación continua:** el progreso del proyecto se revisa semanalmente. Algunas instancias no poseen calificación directa, pero sus avances inciden en la nota de los entregables formales.
   - **Criterios de calificación:** siete instancias de evaluación, con las siguientes ponderaciones:
     - Entregables individuales (EI): 10 %
     - Actividades grupales en aula (AG): 10 %
     - Licitación (L): 15 %
     - Plan de proyecto (PP): 15 %
     - Evaluación económica (EE): 15 %
     - Prototipo inicial (PI): 25 %
     - Compromiso y cumplimiento (CC): 10 %
3. **Organización del proyecto y artefactos**
   - **Tamaño de equipo:** típicamente grupos de 5 a 6 estudiantes; las cargas académicas y extracurriculares pueden diferir entre integrantes.
   - **Cliente del proyecto:** si bien en la planificación del curso se contempla la figura de un cliente, en la práctica este rol puede variar, pudiendo resolverse mediante la definición de un cliente ficticio o incluso con los propios integrantes del equipo asumiendo dicha función.
   - **Plan y continuidad:** se deben definir *historias de usuario* y *criterios de aceptación* que serán desarrollados y evaluados el semestre siguiente en **INF-228**.
   - **Arquitectura del software:** no se exige evidenciar la arquitectura implementada.
4. **Licitación y evaluación externa**
   - **Dos etapas:** una *interna* (equipo docente) y una *externa* (jurado invitado).
   - **Pre-licitación y retrabajo:** los informes de *pre-licitación* pueden ser rechazados, requiriendo reformulación y nueva entrega.
5. **Carga académica y calendario**

- **Carga académica:** por malla, quienes inscriben **INF-360** suelen cursarla junto con **cuatro electivos** y **una asignatura libre** (INF-360 + 5 ramos), lo que supone una **carga considerable**.
- **Duración del semestre:** aproximadamente **4 meses y 2 semanas**.

Se mencionan todas esas consideraciones ya que es muy importante entender el contexto académico en el que el estudiante se encuentra al momento de inscribir la asignatura INF-360.

2.2.2. Taller de Desarrollo de Proyectos de Informática (INF-228)

Descripción General de la Asignatura

La asignatura INF-228, forma parte del eje formativo de Ingeniería Aplicada - Gestión de Proyectos Informáticos. El curso tiene como prerrequisito la aprobación de la asignatura INF-360. El curso tiene un tiempo total de dedicación de 295 horas cronológicas.

El propósito del curso es que los estudiantes integren la metodología de proyectos y apliquen los conocimientos de su plan de estudio para desarrollar una solución informática que aborde un problema real y multidisciplinario. El resultado del proyecto es presentado en una feria informática, siendo el equivalente a un examen final de la asignatura y está abierta tanto a la comunidad interna como externa. La asignatura busca fomentar competencias diferenciadoras como el emprendimiento, la creatividad, la innovación, el autoaprendizaje, el trabajo en equipo y la gestión de proyectos.

Metodología y Contenidos

La metodología de enseñanza se centra en el aprender-haciendo, donde los estudiantes, en equipos formados en la asignatura INF-360, trabajan en un proyecto hasta convertirlo en un prototipo final. Este prototipo debe satisfacer los requerimientos de un cliente real y debe ser presentado en la Feria de software. Los estudiantes deben aplicar técnicas de desarrollo ágil y seguimiento de proyectos, gestionando de manera efectiva el trabajo. Los contenidos temáticos se organizan en las siguientes áreas:

- Gestión de equipos: Abarca la planificación, adquisición, desarrollo y dirección de los recursos humanos del proyecto, incluyendo liderazgo, comunicación, motivación y resolución de conflictos.
- Control del proyecto: Incluye la revisión de la Oficina de Gestión de Proyectos (PMO), el control del alcance, el cronograma y los riesgos. También se aborda el control de calidad, el desarrollo y *testing* del proyecto, y el control de versiones.
- Ejecución y cierre del proyecto: Se centra en la actualización del plan de proyecto, la construcción de hojas de tiempo y fichas de estado, el desarrollo de entregables y la documentación del legado del proyecto.
- Difusión del proyecto: Se enfoca en las estrategias de marketing (imagen corporativa, sitio web y videos promocionales) y la relación con los *stakeholders*. La feria informática es el evento principal para la difusión del proyecto.

Los temas mencionados anteriormente se abordan exclusivamente durante el horario de clases de la asignatura, a través de charlas informativas impartidas por expertos en el área.

Resultados de Aprendizaje y Contribución al Perfil de Egreso

La asignatura busca que los estudiantes logren los siguientes resultados de aprendizaje:

- Aplicar técnicas de manejo de recursos humanos y asignación de roles para el desarrollo de proyectos.
- Gestionar proyectos informáticos multidisciplinarios e innovadores, utilizando metodologías y herramientas reconocidas internacionalmente.
- Construir una solución informática para resolver una problemática real, validada por un cliente.
- Utilizar estrategias de comunicación efectiva y marketing digital para la difusión del proyecto.
- Identificar oportunidades de negocios para mejorar procesos mediante tecnologías informáticas.

El curso contribuye directamente al perfil de egreso en formular, desarrollar y gestionar proyectos informáticos orientados a la innovación y el emprendimiento. Además, fortalece competencias transversales esenciales para el perfil de egreso, tales como la comunicación oral y escrita, la integración y dirección de equipos de trabajo, la autonomía y el pensamiento crítico. También se enfoca en la actualización permanente de competencias, el aseguramiento de la calidad y la manifestación de conductas éticas y de responsabilidad social.

Consideraciones académicas y contextuales (INF-228 — 10.º semestre)

Al inscribir y cursar **INF-228**, conviene atender a cinco ámbitos: (1) *vinculación con cliente*, (2) *estructura y evaluación de la asignatura (sprints y métricas)*, (3) *criterios de calidad usuales no exigidos formalmente*, (4) *organización del proyecto y artefactos*, y (5) *calendario y duración*.

1. **Participación del cliente:** no se exige participación activa; puede existir formalmente sin interactuar ni revisar el desarrollo del producto.
2. **Estructura y evaluación de la asignatura (sprints y métricas)**
   - **Tres sprints:** la evaluación considera las historias de usuario comprometidas por el equipo para cada sprint (historias desarrolladas en INF-360).
   - **Criterios de aceptación y puntos:** cada historia tiene puntos de historia y criterios de aceptación; sólo las historias que cumplen *todos* sus criterios aportan el 100 % de sus puntos.
   - **Cálculo ilustrativo:** si se planifican 3 historias con 10 puntos cada una (30 en total), completar las 3 historias (cumpliendo todos sus criterios) equivale a 30/30, esto es, nota máxima.
   - **Enfoque funcional:** la evaluación se centra en **funcionalidad observable**; la **calidad** del producto no se pondera explícitamente.
3. **Criterios de calidad usuales NO evaluados formalmente**
   - **Calidad interna del código:** legibilidad y mantenibilidad, convenciones, patrones.
   - **Arquitectura y diseño:** modularidad, acoplamiento/cohesión, decisiones arquitectónicas y su documentación.

- **Calidad externa:** usabilidad, rendimiento, confiabilidad, seguridad y portabilidad.
- **Pruebas y aseguramiento:** estrategia y evidencia de pruebas (unitarias, integración, sistema), oráculos, CI/CD y cobertura.
- **Documentación y artefactos:** especificación de requisitos, guías de despliegue/operación y manuales de usuario.

4. **Organización del proyecto y artefactos**
   - **Arquitectura:** no se solicita explícitamente la arquitectura del sistema ni su documentación formal.
   - **Charlas formativas:** se imparten charlas sobre temas relevantes (p. ej., *testing*); son instancias de sensibilización más que de formación técnica profunda.
5. **Calendario y duración**
   - **Cierre anticipado:** el proyecto suele finalizar **1 mes antes** del término del semestre; la ventana efectiva para actividades obligatorias es de aprox. **3 meses y 2 semanas**.

2.3. Definición del problema que aborda ÚneteApp

ÚneteApp es una aplicación móvil desarrollada en el marco de la Feria de Software de la Universidad Técnica Federico Santa María. Su propósito principal es implementar el concepto de prescripción social mediante el uso de técnicas de inteligencia artificial. La aplicación busca dar respuesta a dos problemáticas sociales relevantes: la soledad y el aislamiento social en jóvenes, así como la sobrecarga de los sistemas de salud.

Para abordar estos desafíos, la solución se estructura en dos áreas complementarias. En primer lugar, **ÚneteApp**, dirigida a los jóvenes, cuyo objetivo es facilitar el acceso a actividades y eventos locales, además de permitirles optar a una prescripción social personalizada. En segundo lugar, **ÚneteApp Health System (ÚHS)**, orientada a los profesionales de la salud, que entrega una herramienta de apoyo para la implementación de la prescripción social y que, a su vez, facilita la gestión y el seguimiento de los pacientes derivados a actividades específicas.

Capítulo 3

Marco teórico y Estado del arte

3.1. ¿Qué es *software testing*?

El *software testing* o pruebas de software consiste en la **validación dinámica** de que un **sistema bajo prueba** (SBP) proporciona **comportamientos esperados** en un **conjunto finito** de **casos de prueba** seleccionados adecuadamente de un dominio de ejecución generalmente infinito [36].

Esta es la definición que entrega el SWEBOK (Software Engineering Body of Knowledge), que es un estándar internacional del IEEE (Institute of Electrical and Electronics Engineers) para la ingeniería de software.

Hay que precisar distintas frases que aparecen en la definición anterior para entender en profundidad el concepto de *software testing*. Se explica a continuación a lo que se refiere cada una de las frases basándonos en [36].

Lo primero que hay que mencionar es que el SBP puede ser un programa, producto de software, aplicación, aplicación orientada a servicio, sistema, etc. También se le llama **ítem de prueba** o *test item* [14].

Para llevar a cabo estas pruebas, se utilizan los llamados casos de prueba y se le llama conjunto de pruebas (test suite) a un conjunto de casos de prueba. Un caso de prueba es la especificación de todos los elementos esenciales para la ejecución de una prueba, como los valores de entrada, las condiciones de ejecución y tiempo, el procedimiento y los resultados esperados. Se menciona también que los valores de entrada por sí solos no siempre son suficientes para definir un caso de prueba, ya que el sistema puede reaccionar de manera diferente a la misma entrada dependiendo de su estado o de las condiciones del entorno.

Las entradas y salidas pueden clasificarse ambas en directas o indirectas [20]. Las entradas directas son aquellas que se proporcionan directamente al SBP a través de la interfaz o API, como por ejemplo llamadas a métodos o funciones de otro componente, mensajes enviados a través de un canal de mensajería o datos de entrada en un formulario. Las salidas directas son las respuestas inmediatas del SBP a estas entradas que se muestran en una interfaz, como valores, un mensaje de confirmación, retorno de métodos o funciones, argumentos actualizados pasados por referencia, excepciones generadas o mensajes recibidos en un canal de mensajería. Por otro lado, las entradas indirectas se dan cuando el comportamiento del SBP depende de valores devueltos por otro componente, como por ejemplo resultados de funciones, parámetros de salida actualizados o excepciones generadas. Las salidas indirectas pueden incluir efectos secundarios o cambios

en el estado del sistema que no son inmediatamente visibles pero que son percibidos por otros sistemas o componentes. Por ejemplo, llamadas a funciones de otros componentes, mensajes enviados por canales de mensajería, inserciones de registros en bases de datos o escritura de archivos.

Cabe destacar que cuando se menciona validación dinámica, esta frase hace referencia a la **ejecución del SBP** con un conjunto de pruebas.

Ahora bien, se menciona un conjunto finito de casos de prueba, ya que ejecutar todos los casos de prueba posibles en un sistema, incluso uno simple, podría demorar mucho tiempo, por lo que realizar pruebas exhaustivas no es viable en la práctica. Por esta razón, las pruebas de software se enfocan en una **selección representativa de casos, elegida según distintos criterios**.

Además, no solo se habla de un conjunto finito de casos de prueba, sino de seleccionar adecuadamente los elementos de este conjunto. La tarea de seleccionar los criterios más apropiados para definir los casos de prueba bajo determinadas condiciones constituye un desafío complejo. Para enfrentarlo, es posible aplicar y combinar múltiples enfoques, como el análisis de riesgos, la consideración de los requerimientos del software, la optimización de costos, el cumplimiento de atributos de calidad, la priorización de casos y la capacidad de detección de errores. Las distintas técnicas de prueba varían en la forma en que construyen el conjunto de pruebas, por lo que es fundamental comprender que el uso de diferentes criterios de selección puede dar lugar a niveles de efectividad significativamente distintos.

Una vez seleccionado el conjunto de casos de prueba, es indispensable poder evaluar su resultado. Para cada caso de prueba ejecutado, debe poder evaluarse si el comportamiento observado del SBP coincide con el comportamiento esperado, aunque este análisis no siempre resulte sencillo. Esta comparación puede realizarse en función de las necesidades del usuario (validación), de una especificación formal (verificación), o incluso de expectativas derivadas de requisitos implícitos. Por lo tanto, es esencial definir claramente los resultados esperados para poder juzgar adecuadamente el éxito o fracaso de una prueba. Más adelante se define que es un oráculo de prueba y cómo se relaciona con los resultados esperados.

Hay que destacar que en la definición entregada por el SWEBOK se hace referencia a pruebas dinámicas, lo que significa que el SBP debe estar en ejecución para evaluar que se comporte como se espera. Sin embargo, existen pruebas estáticas las que se realizan sin ejecutar el SBP, como analizar los documentos y el código fuente.

Diversas entidades reconocidas internacionalmente en el ámbito del desarrollo de software, como IBM, GitHub, Atlassian y el ISTQB definen las pruebas de software de la siguiente manera:

- **IBM:** “Las pruebas de software son el proceso de evaluar y verificar que un producto o aplicación de software funcione correctamente, de forma segura y eficiente, de acuerdo con sus requisitos específicos.” [30]
- **GitHub:** “Las pruebas de software son el proceso sistemático de evaluar y verificar aplicaciones de software para garantizar que funcionen correctamente y cumplan con requisitos específicos.” [6]
- **Atlassian:** “Las pruebas de software son un proceso organizacional dentro del desarrollo de software en el cual se verifica que el software crítico para el negocio sea correcto, de calidad y tenga el rendi-

miento adecuado. Las pruebas de software se utilizan para garantizar que los sistemas empresariales esperados y las características del producto se comporten correctamente según lo esperado." [2]

- **ISTQB:** "El proceso dentro del ciclo de vida del desarrollo de software que evalúa la calidad de un componente o sistema y de los productos de trabajo relacionados." [15]

Estas definiciones presentan una notable consistencia entre sí y coinciden con la definición propuesta por el *SWEBOK*, que fue mencionada anteriormente, resaltando que el software testing es una actividad fundamental para garantizar la calidad, el correcto funcionamiento y el cumplimiento de los requisitos de un sistema de software.

3.2. Oráculo

Un oráculo puede estar representado por un agente humano o una herramienta automatizada que, para cada caso de prueba, emite un veredicto de *"aprobado"* o *"fallado"*. No obstante, existen situaciones en las que el oráculo no puede emitir una decisión concluyente; en tales casos, el resultado de la prueba se clasifica como *inconcluso*.

Este oráculo actúa como una referencia confiable que permite decidir si el comportamiento observado del SBP se ajusta a los resultados esperados, entendidos como comportamientos predecibles y observables definidos previamente.

3.3. Error, defecto y falla

Un error puede resultar en un defecto en el producto de software, lo que a su vez puede llevar a una falla del sistema.

Error: Una acción humana que produce un defecto [11].

Defecto: Una imperfección o deficiencia en un producto de trabajo donde no cumple con sus requisitos o especificaciones o perjudica su uso previsto [10].

Falla: Un evento en el que un componente o sistema no cumple con sus requisitos dentro de los límites especificados durante su ejecución [13].

Es de suma importancia aplicar pruebas de software para tener un cierto grado de confianza de que el sistema va a operar como se espera, minimizando la probabilidad de que ocurran fallas en el sistema.

3.4. Creación de casos de prueba

La creación o generación de casos de prueba produce el conjunto de pruebas (test suite) útil para evaluar el SBP con propósitos específicos (por ejemplo, suficiencia, precisión o valoración).

Dado que la generación de casos de prueba es una de las actividades más importantes e intensivas del software testing, normalmente se apoya en enfoques, técnicas y herramientas que permiten automatizar el proceso. El estándar ISO 29119-4:2021 define tres técnicas de diseño de pruebas: basadas en especificación, basadas en estructura y basadas en experiencia [19]. Estas técnicas de diseño de pruebas son complementarias, y su aplicación combinada resulta típicamente en pruebas más efectivas.

Técnicas basadas en especificación (Specification-based testing)

En las pruebas basadas en especificaciones o mejor conocidas como “pruebas de caja negra” (*black-box testing*), la base de pruebas (por ejemplo, requisitos, especificaciones, modelos o necesidades del usuario) se utiliza como la principal fuente de información para diseñar los casos de prueba [8].

Técnicas basadas en estructura (Structure-based testing)

En las pruebas basadas en estructura o mejor conocidas como “pruebas de caja blanca” (*white-box testing*), la estructura del elemento de prueba (por ejemplo, el código fuente o la estructura de un modelo) se utiliza como la principal fuente de información para diseñar casos de prueba [16].

Técnicas basadas en experiencia (Experience-based testing)

En las pruebas basadas en la experiencia, el conocimiento y la experiencia del *tester* se utilizan como la fuente principal de información durante el diseño de casos de prueba [12].

3.5. Priorización y minimización

Se pueden adoptar estrategias adecuadas para la selección o priorización de casos de prueba con el fin de mejorar la eficacia de las pruebas. La priorización de casos de prueba tiene como objetivo definir un orden de ejecución de las pruebas según ciertos criterios (por ejemplo, cobertura, tasa de detección de fallos, similitud y riesgo), de modo que aquellas con mayor prioridad se ejecuten antes que las de menor prioridad.

La minimización del conjunto de casos de prueba busca reducir la cantidad de pruebas adoptando los siguientes principios:

- Evitar casos redundantes ya que estos no aportan valor adicional al proceso de pruebas.
- Cumplir con las restricciones de tiempo y recursos disponibles.
- Mantener la efectividad del conjunto de pruebas, haciendo que estas mantengan su capacidad de detectar fallos.

3.6. Niveles de pruebas

Los niveles de pruebas representan las distintas etapas en las que se evalúa el SBP, desde la verificación de sus componentes individuales hasta la validación final con los usuarios, asegurando su correcto funcionamiento e interoperabilidad. Existen cuatro niveles de pruebas: unitarias, de integración, de sistema y de aceptación [36].

Pruebas unitarias

Las pruebas unitarias verifican el funcionamiento individual de los elementos del SBP que pueden probarse por separado. Dependiendo del contexto, estos elementos pueden ser subprogramas o componentes individuales, un subsistema o una composición de componentes del SBP.

Pruebas de integración

Las pruebas de integración consisten en verificar las interacciones entre los diferentes elementos del SBP, asegurando su correcta interoperabilidad tanto entre ellos como con el entorno externo.

Pruebas de sistema

Las pruebas de sistema consisten en evaluar el comportamiento completo del SBP y verificar que cumpla también con requisitos no funcionales como seguridad, privacidad, velocidad, precisión y confiabilidad.

Pruebas de aceptación

Las pruebas de aceptación verifican que el SBP cumpla con los requisitos y expectativas de los usuarios finales, generalmente con su participación, evaluando aspectos como la usabilidad o la aceptación operativa.

3.7. Tipos de pruebas

Existen múltiples tipos de pruebas de software, cada uno definido por el propósito que persigue y el enfoque con el que se ejecuta. La clasificación de los tipos de pruebas surge principalmente a partir de los objetivos que se buscan alcanzar durante la validación del SBP, lo que permite seleccionar las técnicas más adecuadas para garantizar su correcto funcionamiento y calidad. Estas técnicas pueden ser aplicadas en los distintos niveles de pruebas mencionados anteriormente.

A continuación se definen los tipos de pruebas más relevantes, basados en la clasificación del SWEBOK [36].

Pruebas funcionales

Este tipo de pruebas también se conoce como pruebas de conformidad o de correctitud. En este tipo de pruebas, los casos de prueba se diseñan para verificar que las especificaciones funcionales estén implementadas correctamente. Las especificaciones funcionales describen lo que debe hacer el SBP, es decir, sus funciones y características.

Pruebas de conformidad (Conformance Testing)

Las pruebas de conformidad tienen como objetivo verificar que el SBP cumpla con estándares, reglas, especificaciones, requisitos, diseño, procesos o prácticas.

Pruebas de cumplimiento (Compliance Testing)

Las pruebas de cumplimiento tienen como objetivo demostrar la adherencia del SBP a una ley o regulación [9]. Por lo general, estas pruebas son exigidas por un organismo regulador externo (ej: autoridad gubernamental).

Pruebas de regresión (Regression Testing)

Según la definición reportada en [17], las pruebas de regresión son la "re-ejecución selectiva del SBP para verificar que las modificaciones no hayan causado efectos no deseados y que el SBP aún cumpla con sus requisitos especificados".

En la práctica, este enfoque está diseñado para demostrar que el SBP sigue superando las pruebas que ya había aprobado en un conjunto de pruebas.

En algunos casos, debe realizarse un compromiso entre la seguridad que ofrece ejecutar pruebas de regresión tras cada cambio y los recursos necesarios para realizarlas. Este proceso puede consumir mucho tiempo debido a la gran cantidad de pruebas que podrían ejecutarse.

Las pruebas de regresión son una actividad fundamental en Agile, DevOps, desarrollo guiado por pruebas (TDD) y desarrollo continuo. Generalmente se realizan después de las pruebas de integración y antes del despliegue en producción u operación.

Pruebas no funcionales

Las pruebas no funcionales tienen como objetivo la validación de aspectos no funcionales (como rendimiento, usabilidad o confiabilidad), y se realizan en todos los niveles de prueba.

A continuación se mencionan y definen brevemente algunos tipos de prueba no funcional según [36] y el ISTQB:

- **Pruebas de rendimiento (Performance Testing):** Verifican que el software cumpla los requisitos de rendimiento, evaluando capacidad y tiempos de respuesta.
- **Pruebas de carga (Load Testing):** Evalúan el comportamiento del SBP bajo carga, detectando problemas como interbloqueos, fugas de memoria o pérdida de estabilidad.
- **Pruebas de estrés (Stress Testing):** Llevan el SBP más allá de su capacidad normal para observar su respuesta ante sobrecarga extrema.
- **Pruebas de volumen (Volume Testing):** Evalúan la capacidad de almacenamiento interno y el manejo de grandes volúmenes de datos.
- **Pruebas de conmutación por error (Failover Testing):** Verifican que el SBP continúe operando ante fallos inesperados, reasignando recursos cuando sea necesario.
- **Pruebas de confiabilidad (Reliability Testing):** Determinan si el sistema puede operar de manera continua sin fallos, evaluando estabilidad y corrección de errores.

- **Pruebas de compatibilidad (Compatibility Testing):** Validan que el software funcione correctamente en distintos entornos de hardware, software o versiones.
- **Pruebas de escalabilidad (Scalability Testing):** Evalúan la capacidad del SBP para manejar incrementos en carga, transacciones o volumen de datos.
- **Pruebas de elasticidad (Elasticity Testing):** Verifican que el sistema pueda expandir o reducir rápidamente recursos de cómputo y almacenamiento ante picos de demanda.
- **Pruebas de infraestructura (Infrastructure Testing):** Aseguran la confiabilidad y el rendimiento de los componentes de la infraestructura TI.
- **Pruebas back-to-back (Back-to-Back Testing):** Ejecutan variantes de un mismo programa con idénticas entradas para comparar salidas y detectar discrepancias.
- **Pruebas de recuperación (Recovery Testing):** Verifican la capacidad del sistema para reiniciarse tras fallos o desastres.

3.8. Arquitectura de software

La arquitectura de software es la organización fundamental de un sistema de software, que abarca las estructuras, componentes y relaciones que definen cómo el sistema está compuesto, cómo se comporta y cómo evoluciona a lo largo del tiempo. Implica la toma de decisiones significativas sobre la selección de elementos estructurales, sus interfaces y las interacciones entre ellos, con el fin de satisfacer tanto requisitos funcionales como no funcionales [36, 7, 33]. Esta definición integra los enfoques presentados en las tres fuentes citadas.

Actuando como un plano de desarrollo, la arquitectura de software orienta la construcción del sistema, influye directamente en su calidad, escalabilidad, mantenibilidad y adaptabilidad, y sirve como un medio clave de comunicación entre los distintos actores involucrados en su desarrollo. Asimismo, contar con descripciones claras y precisas de la arquitectura es de suma importancia, ya que constituyen la base para llevar a cabo actividades esenciales como las pruebas de software, el aseguramiento de la calidad y los procesos de certificación [36].

3.9. Software desarrollado en contexto educacional

Es aquel producto concebido, diseñado y desarrollado por estudiantes durante su formación universitaria en cursos de pregrado o posgrado, generalmente como parte de proyectos académicos, trabajos de título, centros de soluciones estudiantiles o cursos de cierre (capstone), con un enfoque en aplicar conocimientos aprendidos y resolver necesidades reales [27]. Lo anterior presupone que el producto desarrollado es evaluado por responsables académicos.

3.10. Testing de software en contexto educacional

Tras revisar el estado del arte sobre las pruebas de software en contextos educativos, específicamente "Software-testing education: A systematic literature mapping" [5], se destacarán los desafíos más importantes que presenta el estudio y se relacionarán estos desafíos con el contexto de la feria de software.

El paper identifica diversos desafíos asociados a las pruebas de software en contextos educativos, y los clasifica en tres categorías: desafíos relacionados con alumnos e instructores, desafíos relacionados con los instructores, y otros desafíos.

Cabe destacar que el estudio se centra en la enseñanza de las pruebas de software, más que en su aplicación en proyectos desarrollados en contextos educativos. No obstante, los desafíos que se mencionan son relevantes para el contexto de la feria de software, ya que su naturaleza es similar.

Desafíos relacionados a alumnos e instructores

- **Las pruebas de software no son bien aceptadas por los estudiantes y existe una baja motivación para aprenderlas**

  El estudio indica que el desafío más común es que a los estudiantes no les gusta aprender sobre pruebas de software. En encuestas realizadas, se repite que las pruebas de software son "tediosas" y "aburridas". También se menciona que los estudiantes con interés en la ingeniería de software tienen más interés en desarrollar software y menos en probar sistemáticamente lo que han creado.

- **Desafíos relacionados a las herramientas de pruebas**

  Para realizar pruebas de software automatizadas, los estudiantes deben aprender nuevas herramientas y bibliotecas. Esto es un desafío, ya que, por lo general, estas herramientas no son fáciles de aprender, especialmente para quienes aún tienen dificultades con los conceptos básicos de la programación. El documento señala que las pruebas de software, y en particular la automatización, es una actividad intensiva en programación, lo que se vuelve complicado para estudiantes con habilidades de programación deficientes. Además, se menciona que las herramientas existentes para técnicas más avanzadas de *testing* suelen ser complejas.

- **Carga cognitiva adicional al aprender pruebas de software**

  Este es un factor que influye en la actitud de los estudiantes hacia las pruebas de software. Se menciona que la programación, por sí sola, ya es demandante, por lo que añadir la carga de aprender sobre pruebas de software es un factor a considerar. Este punto se refiere especialmente a los estudiantes en las etapas iniciales de su carrera, cuando están aprendiendo los fundamentos de la programación.

Desafíos relacionados a instructores

- **Desafíos relacionados al diseño del curso**

  Se mencionan tres puntos a tener en cuenta: la alineación con las necesidades de la industria, la cuestión de la "escala"/complejidad y otros aspectos.

- **Alineación con las necesidades del conjunto de habilidades de la industria:** Se valora la importancia de hacer los cursos de pruebas de software prácticos y alineados con las necesidades de la industria, pero se reconoce el desafío de mantener esta alineación. En algunas universidades, se enseña de manera puramente teórica, lo que genera desinterés en los estudiantes. Una forma de abordarlo es a través de proyectos reales, aunque esto puede ser difícil de conseguir o implementar para los instructores.
- **Cuestión de "escala"/complejidad:** Se menciona que los estudiantes perciben que escribir pruebas para proyectos pequeños es irrelevante. También se argumenta que las pruebas rigurosas son más costosas que beneficiosas para proyectos de pequeña escala, y que los estudiantes no logran comprender plenamente los beneficios de las pruebas de software en estos casos.
- **Otros aspectos a considerar en el diseño del curso:** Se señala que los estudiantes necesitan una retroalimentación constante y concreta sobre cómo mejorar su rendimiento en las pruebas a lo largo del desarrollo del proyecto, en lugar de recibirla solo al final.

- **Restricciones de tiempo y recursos**

  Numerosos estudios mencionan las restricciones de tiempo y recursos que afectan el diseño, la entrega y la evaluación del software. Esto se debe principalmente a que hay tiempo limitado o insuficiente para las pruebas de software, ya que este tema compite con otros en el plan de estudios.

- **Desafíos relacionados con la evaluación del trabajo de los estudiantes**

  Un punto importante que se destaca es la necesidad de evaluar la calidad de las pruebas de software y no solo la corrección del programa en sí. Un enfoque común es la verificación de la cobertura del código de las pruebas, pero esta métrica por sí sola no es una medida satisfactoria de la calidad de las pruebas.

- **Desafíos relacionados con la integración de pruebas de software en otros cursos.**

  El estudio señala que no todos los programas de pregrado de software tienen cursos obligatorios de pruebas de software. Existen muchos tipos de programas con diferentes objetivos y prioridades, y en varios casos, no incluyen cursos de pruebas de software o los ofrecen solo como optativos.

Otros desafíos

Un punto interesante que se menciona es la dificultad de motivar a los estudiantes a probar sus propios programas a fondo, ya que *"cada error encontrado representa un golpe psicológico a su ego como programadores"*.

3.11. Software de salud

Se considera software de salud o *health software* a aquel software que está destinado específicamente a gestionar, mantener o mejorar la salud de personas individuales, o para la prestación de atención médica, o que ha sido desarrollado con el propósito de incorporarse en un dispositivo médico [3].

3.12. Prescripción social

¿Qué es la prescripción social?

La *prescripción social* ha sido abordada por diversas instituciones y estudios, cada uno entregando una perspectiva complementaria sobre su propósito y alcance. A continuación, se presentan algunas definiciones relevantes:

1. World Health Organization (WHO): La prescripción social es un medio para que los trabajadores de la salud conecten a los pacientes con una variedad de servicios no clínicos en la comunidad para mejorar la salud y el bienestar [37].

2. NHS England: Es un enfoque que conecta a las personas con actividades, grupos y servicios en su comunidad para satisfacer necesidades prácticas, sociales y emocionales que afectan su salud y bienestar.

   En la prescripción social, agencias locales como organizaciones benéficas, servicios sociales y de salud derivan a las personas a un *link worker* de prescripción social, quien colabora con la persona para crear un plan de atención personalizado y fomentar el control de su salud y bienestar [23].

3. National Academy for Social Prescribing (NASP): La prescripción social consiste en conectar a las personas con actividades, grupos y apoyos que mejoran la salud y el bienestar. Implica ayudar a que las personas participen en actividades sociales o reciban apoyo práctico, desde servicios de acompañamiento hasta asesoría sobre deudas, grupos de jardinería o clubes de arte, siempre en función de lo que es importante para la persona [22].

4. Artículo de Muhl et al. (2023): La prescripción social se ha definido como "un medio por el cual personas de confianza en entornos clínicos y comunitarios identifican que alguien tiene necesidades sociales relacionadas con la salud que no son médicas, y posteriormente lo conectan con apoyos y servicios no clínicos dentro de la comunidad mediante la coproducción de una prescripción social – una prescripción no médica – para mejorar la salud y el bienestar, y fortalecer las conexiones comunitarias" [21].

5. Stanford Lifestyle Medicine: La prescripción social consiste en que los profesionales de la salud recomienden a sus pacientes involucrarse más en actividades sociales, como llamar por teléfono a familiares, iniciar conversaciones con vecinos o concertar encuentros con amigos para tomar un café [29].

6. OpenLearn (Open University): La prescripción social, también conocida como "derivación comunitaria", implica que un médico, enfermero u otro profesional de la salud derive a la persona a servicios locales no clínicos, como clases de ejercicio, terapia artística u otros recursos comunitarios [25].

A partir de las definiciones anteriores, se puede plantear una conceptualización general de la prescripción social como:

La prescripción social es una estrategia complementaria a la atención sanitaria tradicional que permite derivar a las personas hacia recursos y actividades comunitarias no médicas, como talleres, grupos de apoyo o actividades culturales, con el fin de mejorar su salud física, mental y social. Esta práctica fortalece la autonomía del paciente y promueve la integración comunitaria al abordar necesidades sociales que impactan directamente en el bienestar.

¿Funciona?

La evidencia disponible indica que la prescripción social puede generar beneficios tanto para los pacientes como para los sistemas de salud. Un *mapping review* reciente analizó 87 artículos provenientes de 13 países, identificando 347 resultados únicos, de los cuales 278 corresponden a nivel de paciente y 69 a nivel de sistema de salud [28].

En el caso de los **pacientes**, los beneficios se agrupan en cuatro grandes dimensiones:

- **Salud mental**: Se reportan mejoras en el bienestar mental (19 estudios), aumento de la confianza (16), disminución de la ansiedad (11), reducción de la depresión (11) y menor sensación de soledad (10).
- **Estilo de vida y comportamiento**: Incluye la mejora en el autocontrol de condiciones crónicas (7), activación del paciente para gestionar su salud (5), desarrollo de habilidades (4) y aumento de la independencia (4).
- **Relaciones y conexiones sociales**: Destaca la reducción del aislamiento social (16), el fortalecimiento de la conexión social (5), el incremento del apoyo social (4) y la formación de nuevas amistades (5).
- **Salud física**: Se observan incrementos en la actividad física (11), mejoras en peso e IMC (8) y reducción de la presión arterial (3).

En cuanto a los **beneficios para los sistemas de salud**, el mismo estudio identificó tres categorías principales:

- **Utilización de servicios**: Es la categoría más frecuente a nivel de sistema, con 29 resultados únicos. Incluye cambios en la demanda de servicios de salud mental y atención social, así como en las visitas a médicos de cabecera y servicios de urgencia.
- **Económicos y financieros**: Se identificaron 12 resultados únicos reportados en 14 artículos, destacando el *Social Return on Investment* (SROI) y los ahorros en costos de atención.
- **Uso de medicamentos**: Esta categoría incluye 9 resultados únicos en 10 artículos, principalmente asociados a la reducción del consumo de fármacos y, en particular, de medicamentos psicotrópicos.

Estos hallazgos muestran que la prescripción social tiene el potencial de impactar tanto la salud individual como la eficiencia de los sistemas sanitarios, abarcando dimensiones psicológicas, sociales, físicas y económicas.

Ejemplos de software de prescripción social

- Access Elemental - Social Prescribing Software: https://www.applytosupply.digitalmarketplace.service.gov.uk/g-cloud/services/970429765712985
- Social Rx Connect: https://www.socialrx.co.uk/pages/social-prescribing-software
- Joy - Social Prescribing Software: https://www.thejoyapp.com/not-in-use/social-prescribing#our-products
- Cybermedia - Social Prescribing Software: https://www.cyber-media.co.uk/offer/social-prescribing-software/
- Help At Hand: https://hand.community/

Capítulo 4

Levantamiento de la arquitectura y proceso de pruebas

4.1. Arquitectura de ÚneteApp

ÚneteApp fue desarrollada en el marco de la Feria de Software, instancia en la que junto con el equipo, se tomaron decisiones arquitectónicas, se definieron las historias de usuario y criterios de aceptación implementados en la aplicación. Sin embargo, en dicho contexto no se elaboraron documentos formales que describieran la arquitectura del software, ya que no constituía un requisito.

En cuanto a la arquitectura tecnológica implementada, el *frontend* de la aplicación fue desarrollado en *React Native*. Para el almacenamiento y la persistencia de datos se utilizó una base de datos relacional (*PostgreSQL*) en conjunto con *Azure blob Storage*. El *backend* fue implementado en *Python*, construyéndose una API con *FastAPI*. Finalmente, para el módulo de prescripción social se entrenó un modelo de aprendizaje automático encargado de recomendar actividades a los usuarios.

Los componentes principales del sistema, en particular la base de datos relacional y el *backend*, fueron containerizados y orquestados mediante *Docker Compose*, definiéndose una red interna que permitió la comunicación segura y aislada entre dichos servicios. La aplicación móvil, por su parte, se comunica con el *backend* a través de protocolos estándar *HTTP*, garantizando la correcta interacción entre cliente y servidor.

A continuación, se presenta el levantamiento de la arquitectura general de ÚneteApp, elaborada a partir de los informes y documentos generados durante INF-360 e INF-228, así como de conversaciones con los integrantes del equipo y el análisis del código fuente. Posteriormente, se detallan aspectos específicos del funcionamiento, estado y flujo de la aplicación con el fin de facilitar y orientar el diseño de las pruebas de software. Todos los diagramas que se levantaron en esta sección fueron realizados utilizando la herramienta de desarrollo *Visual Paradigm* [1] y el lenguaje de modelado UML estandarizado por OMG [2] [24].

[1] Para más información visitar https://www.visual-paradigm.com/
[2] Para más información visitar https://www.omg.org/spec/UML/

Despliegue de ÚneteApp

Para comprender la arquitectura de la aplicación, se identificaron y documentaron las relaciones entre los elementos lógicos y físicos del sistema, así como los activos de tecnología de la información asignados a ellos. Este levantamiento se realizó mediante una representación gráfica que permite visualizar los componentes principales y sus interacciones. En la figura 4.1 se presenta dicha representación para ÚneteApp.

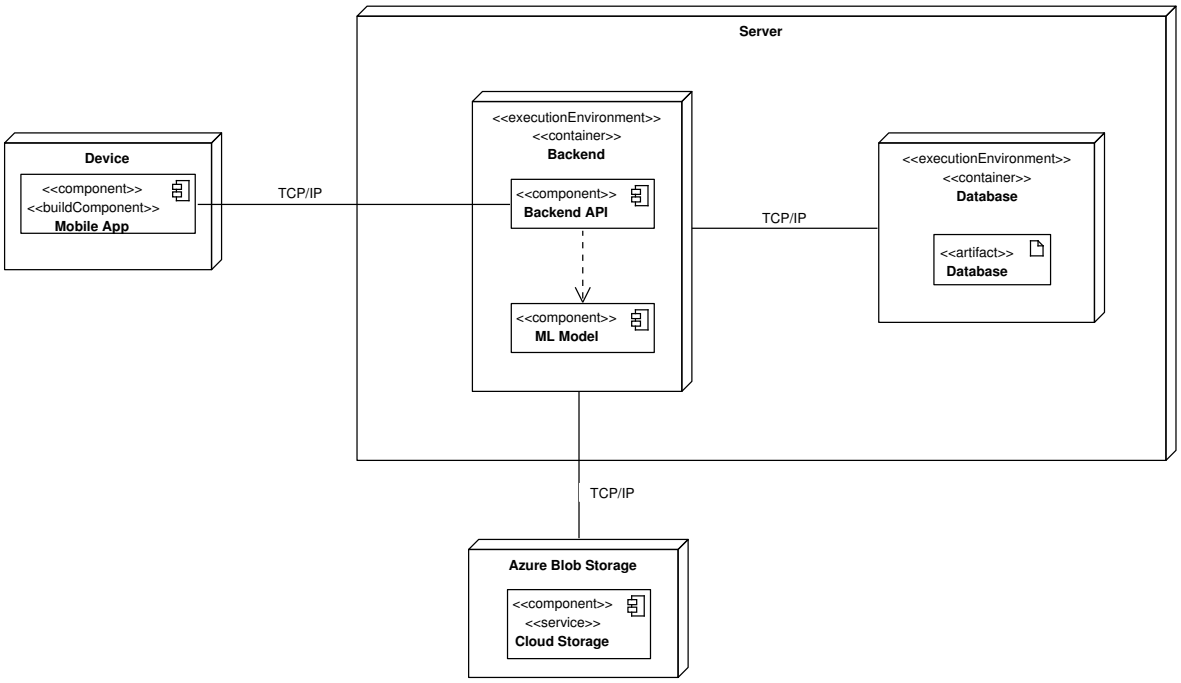


Figura 4.1: Diagrama de despliegue de ÚneteApp

Como se puede ver en la figura 4.1 el despliegue de ÚneteApp se compone de cuatro nodos principales: El nodo *Backend* incluye una API y un modelo de inteligencia artificial que se encarga de entregar la prescripción social, el cual presenta una conexión al nodo *Azure Blob Storage* para la gestión de imágenes. El nodo *Database* almacena toda la información de la aplicación, mientras que el *frontend* corresponde al nodo *Device* que representa la aplicación móvil, la cual interactúa con el backend para presentar la información al usuario.

Historias de usuario desarrolladas para ÚneteApp

A continuación, se presentan las historias de usuario que fueron desarrolladas en INF-360 e implementadas en INF-228 para ÚneteApp y fueron mostradas en el prototipo final durante la Feria de Software. Estas historias de usuario definen las funcionalidades clave de la aplicación. Cada historia de usuario incluye una breve descripción, los criterios de aceptación y la cantidad de puntos de historia que fueron asignados a cada una. Por simplicidad se adjuntaron solamente el nombre de la historia de usuario, el número de puntos de historia y la prioridad de la historia (Crítico, Alto, Medio y Bajo).

Historias de usuario documentadas para el sprint 1 en el anexo B.2, sprint 2 en el anexo B.3, y sprint 3 en el anexo B.4.

Es importante mencionar que las historias de usuario que fueron implementadas en el sprint 3 se clasificaron con una prioridad media, a pesar de ser críticas para el funcionamiento de ÚHS.

Levantamiento de casos de uso de ÚneteApp

A partir de las historias de usuario implementadas en ÚneteApp, se describieron casos de uso y a partir de ellos se representaron los casos de uso en diagramas UML destinados a tales efectos, uno para ÚneteApp y otro para ÚHS, los cuales ilustran las interacciones entre los diferentes actores y las funcionalidades del sistema.

Se muestra a continuación en la figura 4.2 el diagrama de casos de uso [3] para ÚHS donde se identifican dos actores principales "Health Professional" y "Patient" y los casos de uso asociados a cada uno de ellos:

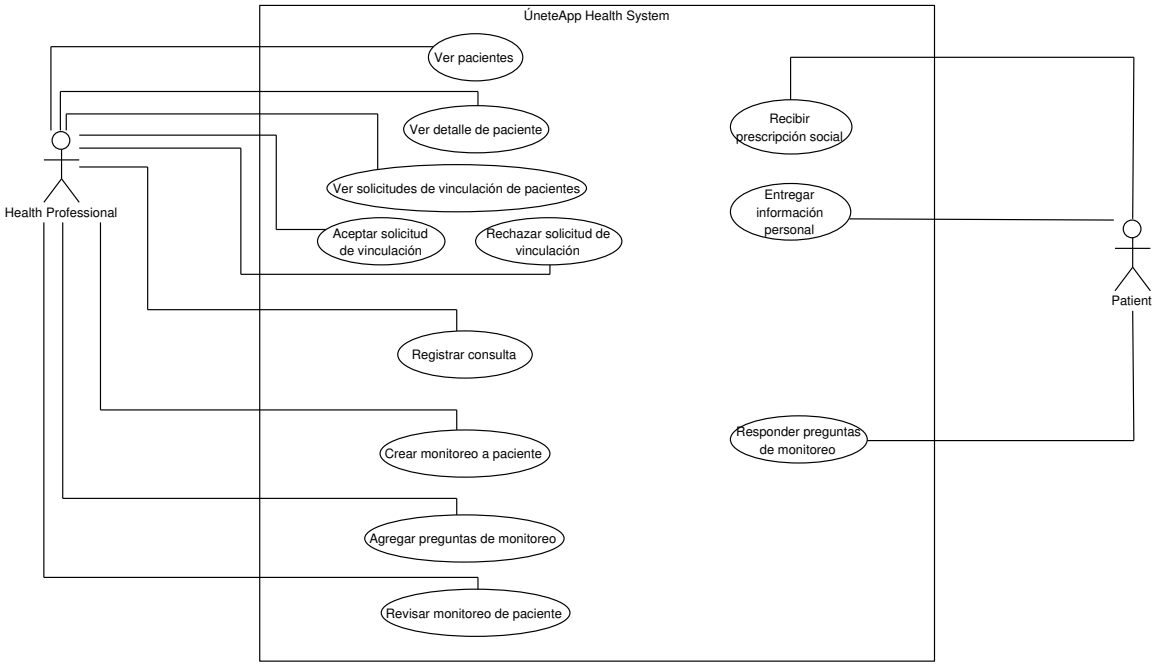


Figura 4.2: Diagrama de casos de uso de ÚneteApp Health System

En el anexo B.1, se presenta el diagrama de casos de uso para ÚneteApp.

Priorización de casos de uso

Dado el tiempo limitado no es posible realizar las pruebas para todos los casos de uso levantados a partir de las historias de usuario implementadas. Por lo tanto, se priorizaron los casos de uso basándonos en la prioridad de las historias (crítico, alto, medio y bajo, donde crítico es la de mayor prioridad) que fue definida en INF-360. Se aplica el proceso de pruebas de software a los casos de uso clasificados como críticos corres-

[3] Diagrama UML que muestra relaciones entre casos de uso y actores. Más información en [35]

pondientes a ÚHS, especificamente a los 4 primeros casos de uso asociados al actor 'Health Professional' y al primero asociado al actor 'Patient'.

A continuacion, en la tabla 4.1 se muestra la priorizacion de los casos de uso donde los clasificados como **críticos** son los considerados para el proceso de pruebas de software:

Caso de uso	**Actor involucrado**	**Prioridad**
(UHS) Ver pacientes	Health Professional	Crítico
(UHS) Registrar consulta	Health Professional	Crítico
(UHS) Aceptar solicitud de vinculación	Health Professional	Crítico
(UHS) Rechazar solicitud de vinculación	Health Professional	Crítico
(UHS) Ver detalle de paciente	Health Professional	Crítico
(UHS) Ver solicitudes de vinculación de pacientes	Health Professional	Crítico
(UHS) Crear monitoreo a paciente	Health Professional	Crítico
(UHS) Agregar preguntas de monitoreo	Health Professional	Crítico
(UHS) Revisar monitoreo de paciente	Health Professional	Crítico
(UHS) Recibir prescripción social	Patient	Crítico
(UHS) Entregar información personal	Patient	Crítico
(UHS) Responder preguntas de monitoreo	Patient	Crítico
Entregar información personal	General user	Crítico
Registrar experiencia de participación en actividad	General user	Crítico
Registrar asistencia a actividad	General user	Crítico
Recibir prescripción social	General user	Crítico
Ver actividades	General user	Alto
Filtrar actividad	General user	Alto
Buscar actividad	General user	Alto
Ver detalle de actividad	General user	Alto
Informar nueva actividad	General user	Alto
Reportar información errónea de actividad	General user	Alto
Revisar actividad informada	Moderador	Alto
Crear actividad	Organizador	Alto
Publicar actividad	Organizador	Alto
Editar actividad	Organizador	Alto
Eliminar actividad	Organizador y Moderador	Alto

Tabla 4.1: Priorización de casos de uso para pruebas de software

Fuente: Elaboración propia

Vista dinámica del sistema

Se levantaron diagramas de secuencia [4] del sistema para la mayoría de los casos de uso, con el fin de entender mejor el flujo de la aplicación y cómo interactúan los diferentes componentes. Estos diagramas fueron desarrollados a partir de la interacción directa con la aplicación desarrollada haciendo un seguimiento de las llamadas al backend y las respuestas del mismo, así como la interación con las interfaces de usuario. Los diagramas de secuencia se encuentran en el siguiente drive aquitectura ÚneteApp.

Como ejemplo, se muestra el diagrama de secuencia del caso de uso para ÚHS "Ver pacientes".

[4] Diagrama UML que muestra la interacción entre objetos. Más información en [34]

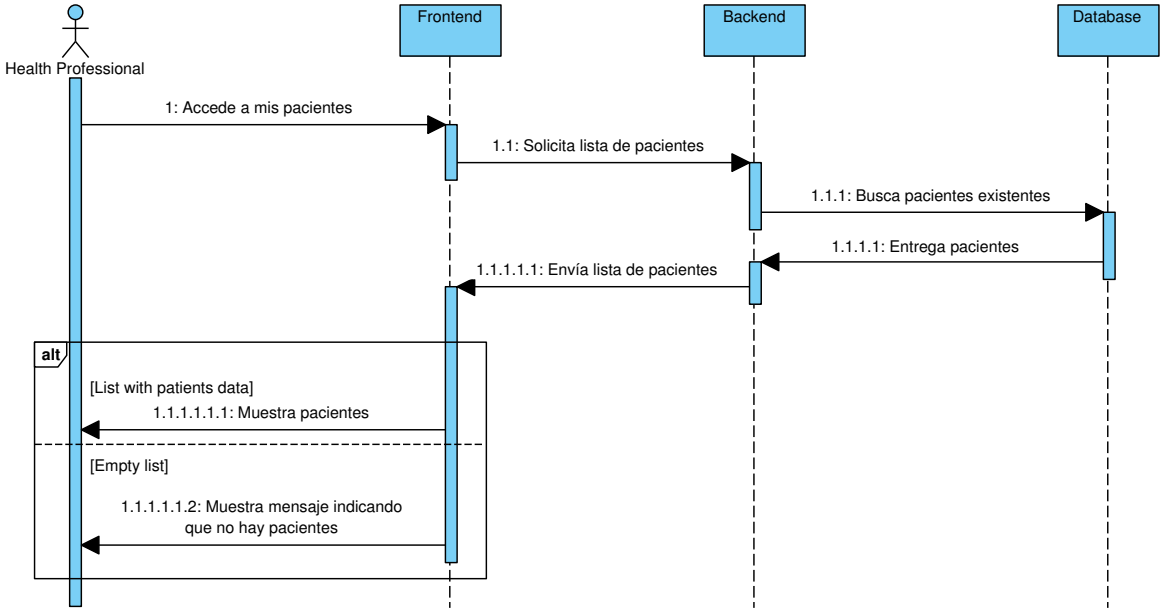


Figura 4.3: Diagrama de Secuencia del caso de uso “Ver paciente” de ÚHS

Vista dinámica del sistema con enfoque en diseño de pruebas

Ahora que se cuenta con un entendimiento más detallado del sistema gracias al levantamiento de la arquitectura de ÚneteApp y por sobre todo a través de los diagramas de secuencia levantados, se diseñaron diagramas de secuencia de componentes del sistema con un enfoque en el diseño de pruebas. Es de suma importancia contar con estos diagramas, ya que de estos diagramas se derivaron los ítems de prueba y bases de prueba que se utilizaron en la primera etapa del proceso de pruebas dinámicas.

Los diagramas presentados a continuación especifican en detalle los endpoints involucrados, junto con sus respectivas entradas y salidas esperadas. El propósito de estos diagramas es proporcionar una referencia precisa para la implementación del proceso de pruebas, asegurando la cobertura integral de todos los endpoints asociados a cada caso de uso, dado que estos serán considerados como los ítems de prueba.

Como ejemplo, se presenta el diagrama asociado al caso de uso “Ver paciente” 4.4.

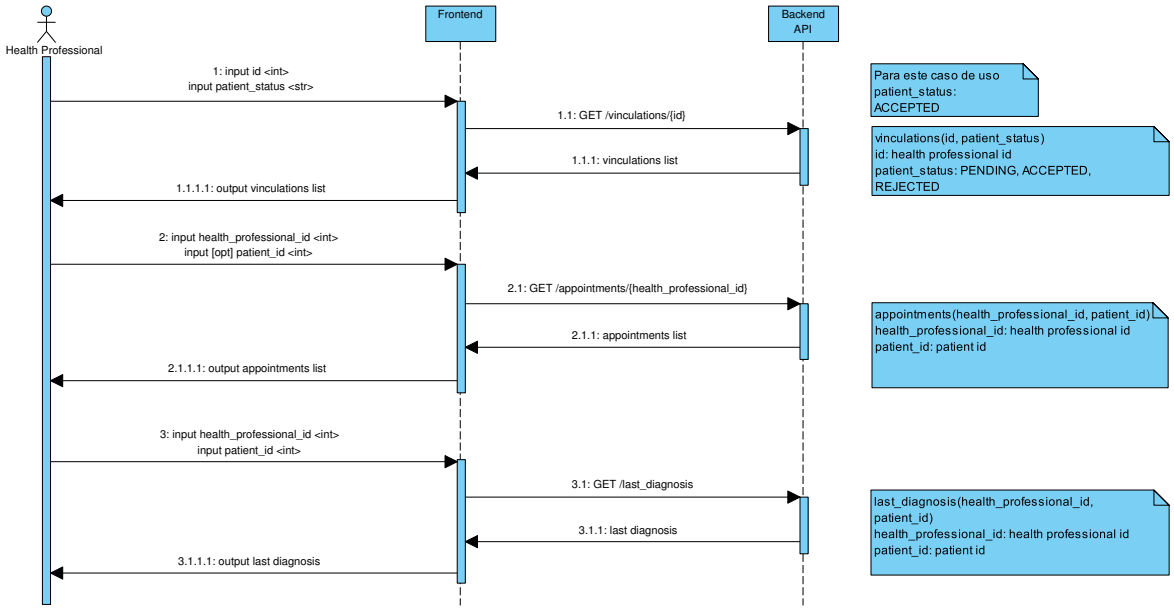


Figura 4.4: Diagrama de secuencia de componentes del sistema del caso de uso “Ver pacientes” de ÚHS

Se puede observar en 4.4 que el caso de uso involucra tres endpoints: `GET /vinculations/{id}`, `GET /appointments/{health_professional_id}` y `GET /last_diagnosis`. También se pueden ver las entradas y salidas esperadas asociadas a cada endpoint.

Se levantaron estos modelos únicamente para los casos de uso de **ÚHS**. Esto se debe a que ÚHS fue desarrollado en el último sprint de INF-228, en un escenario en el que el equipo se encontraba contra el tiempo para implementar las historias de usuario asociadas. Debido a esto, dichas historias no fueron clasificadas con prioridad *crítica*, aunque en la realidad son de gran relevancia para el correcto funcionamiento de ÚHS.

Además, la implementación en ese último sprint no alcanzó el mismo nivel de detalle ni rigurosidad que el trabajo realizado en los sprints anteriores, ya que se desarrolló de manera más apresurada. Por estas razones, se ha decidido elaborar diagramas de componentes de sistema orientados al diseño de pruebas para ÚHS, con el fin de cubrir aquellas áreas críticas que, por las condiciones del desarrollo, no recibieron la misma atención en etapas previas.

4.2. Enfoque utilizado para el proceso de pruebas

Se siguió el proceso sugerido por la ISO 29119-2:2021 acerca de *“Dynamic test processes”* 1.1. Este proceso cuenta con las siguientes etapas y subetapas:

1. Proceso de diseño e implementación
    a) Crear modelo de prueba (TD1)
    b) Identificar ítems de cobertura de prueba (TD2)
    c) Derivar casos de prueba (TD3)

d) Crear procedimientos de prueba (TD4)

2. Proceso de manejo de datos y entorno de pruebas

   a) Establecer el entorno de prueba (ED1)

   b) Preparar datos de prueba (ED2)

   c) Mantener el entorno de prueba (ED3)

   d) Mantener datos de prueba (ED4)

3. Proceso de ejecución

   a) Ejecutar procedimientos de prueba (TE1)

   b) Comparar resultados de prueba (TE2)

   c) Registrar la ejecución de la prueba (TE3)

4. Proceso de reporte de incidentes de pruebas

   a) Analizar los resultados de la prueba (IR1)

   b) Crear/actualizar el reporte de incidente (IR2)

Se diseñaron las pruebas usando técnicas de diseño basado en especificación (caja negra) centrandose en verificar el correcto funcionamiento (pruebas funcionales) de los **endpoints** involucrados en los casos de uso prioritarios (categorizados como críticos de ÚHS), basandonos en los diagramas de secuencia de componentes del sistema desarrollados en la seccion 4.1. Solamente se realizaron pruebas unitarias, considerando al endpoint como item de prueba para asegurar que cada uno funcione correctamente de manera aislada.

4.3. Entorno y herramientas para pruebas dinámicas

Previo al desarrollo del proceso de pruebas dinámicas, es necesario detallar el conjunto de herramientas y tecnologías utilizadas como soporte en la ejecución de las pruebas. La correcta definición de este entorno permite garantizar la reproducibilidad de los resultados y la trazabilidad de los artefactos generados durante la aplicación del proceso formal de pruebas de software.

Se han considerado los siguientes elementos a utilizar para esta memoria:

- **Robot Framework**: se empleó como marco para la automatización de pruebas, dada su flexibilidad, expresividad y amplia adopción en la industria. Considerando que el sistema bajo prueba exponía su funcionalidad principal a través de un backend desarrollado en *FastAPI*, Robot Framework resultó una alternativa idónea para la automatización de pruebas sobre APIs, permitiendo además la generación de reportes y registros de ejecución estandarizados en formatos *HTML* y *XML*, lo cual fue de utilidad para las etapas 3 y 4 del proceso.

- **Docker**: se utilizó para desplegar un entorno de pruebas aislado y reproducible, garantizando que los casos de prueba pudieran ejecutarse bajo condiciones controladas. Esta decisión fue especialmente relevante en un contexto donde intervenían múltiples componentes (frontend, backend, base de datos y servicios externos como Azure Blob Storage), ya que permitió mantener la coherencia de las configuraciones y reducir los riesgos de incompatibilidad, aspecto fundamental para la etapa 2 del proceso. Se implementó una red de Docker con tres servicios principales (véase B.1): backend-test, postgres-test y test. Estos servicios se comunicaron entre sí a través de una red interna de Docker, y se habilitó la comunicación con los contenedores desde el host mediante el mapeo de puertos. De este modo, fue posible ejecutar la aplicación con un backend y una base de datos específicos para pruebas, asegurando la ejecución de los casos en un ambiente controlado y dedicado.

- **Base de datos de prueba**: se preparó una base de datos específica para la ejecución de las pruebas, derivada de la estructura y los datos utilizados durante la presentación de la aplicación en la Feria de Software. Este enfoque permitió disponer de información realista y representativa, asegurando al mismo tiempo que los datos de prueba pudieran ser controlados, reiniciados y aislados de la base de datos de producción. Tal como se mencionó previamente, se creó un contenedor de Docker con una instancia de PostgreSQL destinada exclusivamente a pruebas, la cual constituyó una copia de la base de datos utilizada en la Feria de Software.

Para la ejecución de las pruebas dinámicas, se utilizó una copia específica del proyecto basada en la versión más reciente de la aplicación. Esta copia constituyó el entorno base sobre el cual se llevaron a cabo todos los casos de prueba, garantizando la consistencia y trazabilidad de los resultados obtenidos.

Cabe señalar que, si bien en esta memoria no se aborda la integración continua, la elección de estas herramientas también posibilita en un trabajo futuro la integración con pipelines CI/CD (por ejemplo, GitHub Actions o GitLab CI), de modo que las pruebas puedan ejecutarse automáticamente en cada despliegue del sistema.

4.4. Ítems de prueba identificados

De acuerdo con los diagramas de secuencia de componentes de sistema, es decir, los diagramas enfocados a pruebas levantados en la sección anterior, se identificaron los endpoints involucrados en cada caso de uso prioritario (crítico) de ÚHS. Cada uno de los endpoints, así como sus entradas y salidas, fueron utilizados como *base de prueba* para el diseño de las pruebas. Cada endpoint se considera como una *unidad*, por lo que se realizarán pruebas unitarias para cada uno de ellos.

Endpoints identificados en diagramas de secuencia de componentes del sistema levantados:

- **Ver pacientes**:
  - `GET /vinculations/{id}`
  - `GET /appointments/{health_professional_id}`
  - `GET /last_diagnosis`
- **Ver detalle de paciente**:

- `GET /user/{id}`
- `GET /appointments/{health_professional_id}`
- `GET /last_diagnosis`
- `GET /pacient_monitoring_question`

- **Ver solicitudes de vinculación de pacientes**:
  - `GET /vinculations/{id}`
- **Aceptar solicitud de vinculación**:
  - `PUT /vinculations/{id}`
- **Rechazar solicitud de vinculación**:
  - `PUT /vinculations/{id}`
- **Registrar consulta**:
  - `POST /appointment`
- **Crear monitoreo a paciente**:
  - `POST /pacient_monitoring_question`
- **Agregar preguntas de monitoreo**:
  - `POST /monitoring_question/{patient_id}`
- **Revisar monitoreo de paciente**:
  - `GET /pacient_monitoring_question`
- **Recibir prescripción social**:
  - `GET /prescription/status/{patient_id}`
  - `POST /prescription/{patient_id}`
- **Entregar información personal**:
  - `GET /question`
  - `GET /prescription/status/{patient_id}`
  - `GET /prescription/pending_question/{patient_id}`
  - `POST /user_answer`
- **Responder a preguntas de monitoreo**:
  - `GET /pacient_monitoring_question`
  - `POST /monitoring_user_answer`

Dado que algunos endpoints se encuentran compartidos entre distintos casos de uso, a continuación se presentan sin repetir. Los endpoints resaltados en negro corresponden a los considerados dentro del proceso de pruebas dinámicas:

1. **`GET /vinculations/{id}`**
2. **`GET /appointments/{health_professional_id}`**
3. **`GET /last_diagnosis`**
4. `GET /user/{id}`
5. `GET /pacient_monitoring_question`
6. **`PUT /vinculations/{id}`**
7. **`POST /appointment`**
8. `POST /pacient_monitoring_question`
9. `POST /monitoring_question/{patient_id}`
10. **`GET /prescription/status/{patient_id}`**
11. **`POST /prescription/{patient_id}`**
12. `GET /question`
13. `GET /prescription/pending_question/{patient_id}`
14. `POST /user_answer`
15. `POST /monitoring_user_answer`

Debido al elevado número de endpoints involucrados, se expone en detalle el procedimiento completo únicamente para el primer endpoint, `GET /vinculations/{id}`, correspondiente a la etapa inicial de **diseño e implementación de las pruebas**. Las tres etapas restantes del proceso de pruebas se aplican de manera común al resto de los endpoints.

Cabe señalar que el proceso de pruebas dinámicas se realizó exclusivamente sobre los endpoints derivados de los cuatro primeros casos de uso asociados al actor *"Health Professional"*, así como sobre el primer caso de uso correspondiente al actor *"Patient"*. Todos ellos fueron categorizados como críticos en la tabla 4.1. El procedimiento completo puede revisarse en el apéndice A.

4.5. Primera etapa: Diseño e implementación de las pruebas

Base de prueba

Se considera como ítem de prueba el endpoint `GET /vinculations/{id}` que se puede ver en la figura asociada a la vista dinámica del caso de uso *"Ver pacientes"* (figura 4.4). A partir del ítem de prueba se

deriva la base de prueba:

El endpoint recibe dos parámetros como entrada: `id` y `patient_status`. El parámetro de ruta `id` corresponde al identificador asociado al profesional de la salud. Se considerarán tanto ruts como números de registro médico, ya que ambos pueden ser utilizados como identificadores de profesionales de la salud. El parámetro de consulta `patient_status` corresponde al estado de los pacientes que se desean obtener y puede tomar los valores `ACCEPTED`, `PENDING` o `REJECTED`.

La salida esperada corresponde a una lista de vinculaciones, correspondientes a las vinculaciones del profesional de la salud con los pacientes cuyo `status` coincide con el valor entregado como parámetro de consulta `patient_status`. En caso de que no existan vinculaciones que cumplan con dicho criterio, la salida esperada es una lista vacía.

Si se detecta una entrada inválida, la salida esperada es un mensaje de error en conjunto del código de estado HTTP correspondiente.

Se utilizará la técnica de particiones de equivalencia en conjunto con valores de frontera para derivar los ítems de cobertura.

El formato de un elemento vinculación de la lista de vinculaciones se puede ver en B.2.

Cobertura requerida del ítem de prueba

Cobertura completa de las particiones de equivalencia y valores de frontera identificados.

Modelo de pruebas (TD1)

Se desarrolla un diagrama 4.5 que representa las particiones de equivalencia para el endpoint.

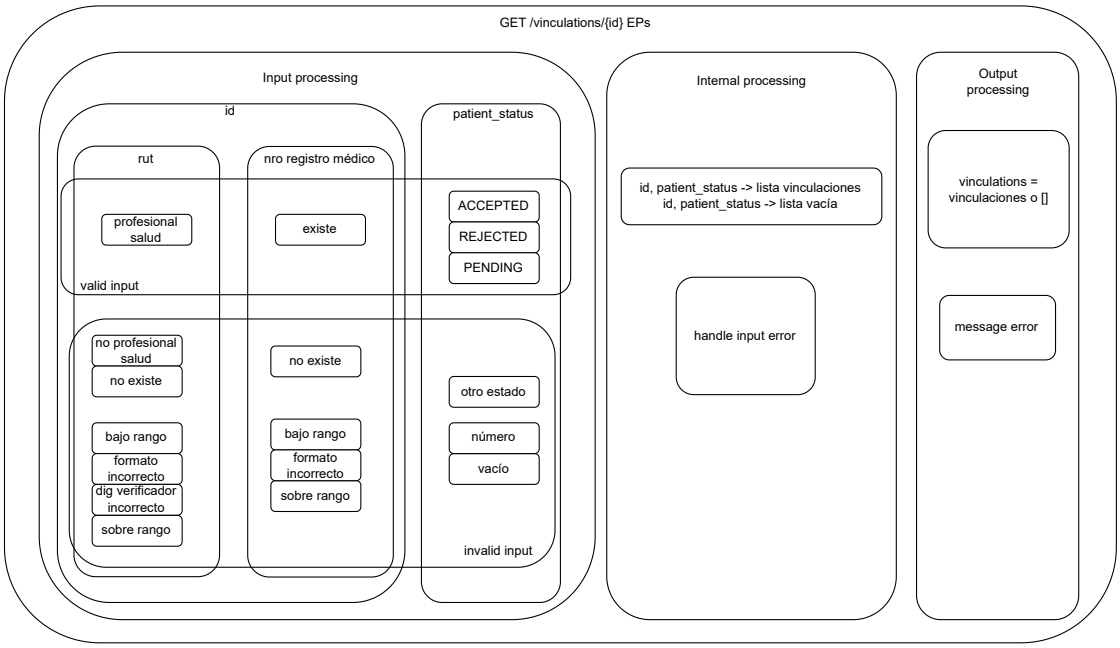


Figura 4.5: Particiones de equivalencia del endpoint GET /vinculations/{id}

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama 4.5. Del procesamiento de la entrada:

- TESTCOVER1: id como rut correspondiente a un profesional de la salud (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: id como número de registro médico existente (10 ≤ nro registro médico ≤ 999.999).
- TESTCOVER3: id como rut no correspondiente a un profesional de la salud.
- TESTCOVER4: id como rut no existente.
- TESTCOVER5: id como rut bajo el rango de ruts válidos (id ≤ 999.999).
- TESTCOVER6: id como rut sobre el rango de ruts válidos (100.000.000 ≤ id).
- TESTCOVER7: id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER8: id como rut con formato incorrecto (letras).
- TESTCOVER9: id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER10: id como rut con dígito verificador incorrecto.
- TESTCOVER30: id como número de registro médico no existente.

- TESTCOVER11: id como número de registro médico bajo el rango de registros médicos válidos (id ≤ 9).
- TESTCOVER12: id como número de registro médico sobre el rango de registros médicos válidos (1.000.000 ≤ id).
- TESTCOVER13: id como número de registro médico con formato incorrecto (caracteres especiales).
- TESTCOVER14: id como número de registro médico con formato incorrecto (letras).
- TESTCOVER15: patient_status con valor permitido 'ACCEPTED'.
- TESTCOVER16: patient_status con valor permitido 'REJECTED'.
- TESTCOVER17: patient_status con valor permitido 'PENDING'.
- TESTCOVER18: patient_status con otro estado (ej: 'EN PROCESO').
- TESTCOVER19: patient_status como número.
- TESTCOVER20: patient_status como número decimal.
- TESTCOVER21: patient_status como cadena vacía.

De los ítems de cobertura 1, 2, 5, 6, 11 y 12 se identifican los siguientes valores de frontera:

- BOUND1-inf: id como rut dentro del límite inferior (id = 1.000.000).
- BOUND1-inf-out: id como rut fuera del límite inferior (id = 999.999).
- BOUND1-sup: id como rut dentro del límite superior (id = 99.999.999).
- BOUND1-sup-out: id como rut fuera del límite superior (id = 100.000.000).
- BOUND2-inf: id como número de registro médico dentro del límite inferior (id = 10).
- BOUND2-inf-out: id como número de registro médico fuera del límite inferior (id = 9).
- BOUND2-sup: id como número de registro médico dentro del límite superior (id = 999.999).
- BOUND2-sup-out: id como número de registro médico fuera del límite superior (id = 1.000.000).
- BOUND5: id como rut dentro del rango de ruts válidos (id = 999.999).
- BOUND5-out: id como rut fuera del rango de ruts válidos (id = 1.000.000).
- BOUND6: id como rut dentro del rango de ruts válidos (id = 99.999.999).
- BOUND6-out: id como rut fuera del rango de ruts válidos (id = 100.000.000).
- BOUND11: id como número de registro médico dentro del rango de registros médicos válidos (id = 9).
- BOUND11-out: id como número de registro médico fuera del rango de registros médicos válidos (id = 10).

- BOUND12: id como número de registro médico dentro del rango de registros médicos válidos (id = 999.999).
- BOUND12-out: id como número de registro médico fuera del rango de registros médicos válidos (id = 1.000.000).

Del procesamiento interno:

- TESTCOVER22: lista con vinculaciones aceptadas es inducida.
- TESTCOVER23: lista con vinculaciones rechazadas es inducida.
- TESTCOVER24: lista con vinculaciones pendientes es inducida.
- TESTCOVER25: lista de vinculaciones vacía es inducida.
- TESTCOVER26: mensaje de error es inducido.

Del procesamiento de la salida:

- TESTCOVER27: salida como lista de vinculaciones y código de estado HTTP 200.
- TESTCOVER28: salida como lista vacía y código de estado HTTP 200.
- TESTCOVER29: salida como mensaje de error y código de estado HTTP 4xx.

Adicionalmente, los siguientes formatos para la entrada `id` como rut son considerados:

- TESTCOVER31: rut valido con formato puntos y guion 'XX.XXX.XXX-X'.
- TESTCOVER32: rut valido con formato sin puntos y con guion 'XXXXXXXX-X'.
- TESTCOVER33: rut valido con formato sin puntos ni guion 'XXXXXXXXX'.
- TESTCOVER34: rut valido con formato con puntos y sin guion 'XX.XXX.XXXX'.

Derivación de los casos de prueba (TD3)

Ahora habiendo identificado los ítems de cobertura, se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se deriva un conjunto de casos de prueba siguiendo la restricción de que la cardinalidad del conjunto es mínima, con tal de pasar por todas las particiones de equivalencia y valores de frontera, es decir, cada caso de prueba cubre la mayor cantidad de ítems de cobertura posibles.

Se muestra en la tabla 4.2 los casos de prueba derivados de los ítems de cobertura. Todos los ítems de cobertura identificados en TD2 estan presentes en los casos de prueba, logrando un 100 % de cobertura.

Caso de prueba	EP involucrada	Entrada (id)	Entrada (patient_status)	Ítem de cobertura	Salida esperada
1	entrada válida: id y patient_status	18.475.065-1	”ACCEPTED”	TESTCOVER1, TESTCOVER15, TESTCOVER22, TESTCOVER27, TESTCOVER31	lista de vinculaciones aceptada(HTTP 200)
2	entrada válida: id y patient_status	550129	”REJECTED”	TESTCOVER2, TESTCOVER16, TESTCOVER23	lista de vinculaciones rechazadas (HTTP 200)
3	entrada inválida: id	20957173-0	”PENDING”	TESTCOVER3, TESTCOVER17, TESTCOVER26, TESTCOVER29, TESTCOVER 32	’Health professional not found’ (HTTP 404)
4	entrada inválida: id	22.208.530-6	”ACCEPTED”	TESTCOVER4, TESTCOVER15, TESTCOVER26, TESTCOVER29	’Health professional not found’ (HTTP 404)
5	entrada inválida: id	999.999-K	”REJECTED”	TESTCOVER5, BOUND1-inf-out, BOUND5	’Invalid id: length violation’ (HTTP 422)
6	entrada inválida: id	1.000.000-9	”PENDING”	TESTCOVER5, BOUND1-inf, BOUND5-out	’Health professional not found’ (HTTP 404)
7	entrada inválida: id	100.000.000-7	”ACCEPTED”	TESTCOVER6, BOUND1-sup-out, BOUND6	’Invalid id: length violation’ (HTTP 422)
8	entrada inválida: id	99.999.999-9	”REJECTED”	TESTCOVER6, BOUND1-sup, BOUND6-out	’Health professional not found’ (HTTP 404)
9	entrada inválida: id	12.110.471*7	”PENDING”	TESTCOVER7	’Invalid id: unexpected characters’ (HTTP 400)
10	entrada inválida: id	12A110471-7	”ACCEPTED”	TESTCOVER8	’Invalid id: unexpected characters’ (HTTP 400)
11	entrada inválida: id	12110471	”REJECTED”	TESTCOVER9	’Invalid id: missing check digit’ (HTTP 400)
12	entrada inválida: id	20.957.173-2	”PENDING”	TESTCOVER10	’Invalid id: check digit’ (HTTP 400)
13	entrada inválida: id	9	”ACCEPTED”	TESTCOVER11, BOUND2-inf-out, BOUND11	’Invalid id: length violation’ (HTTP 422)
14	entrada inválida: id	10	”REJECTED”	TESTCOVER11, BOUND2-inf, BOUND11-out	’Health professional not found’ (HTTP 404)
15	entrada inválida: id	1000000	”PENDING”	TESTCOVER12, BOUND2-sup-out, BOUND12	’Invalid id: length violation’ (HTTP 422)
16	entrada inválida: id	999999	”ACCEPTED”	TESTCOVER12, BOUND2-sup, BOUND12-out, TESTCOVER30	’Health professional not found’ (HTTP 404)

Caso de prueba	EP involucrada	Entrada (id)	Entrada (patient_status)	Ítem de cobertura	Salida esperada
17	entrada inválida: id	#685848	"REJECTED"	TESTCOVER13	'Invalid id: unexpected characters' (HTTP 400)
18	entrada inválida: id	68A5848	"PENDING"	TESTCOVER14	'Invalid id: unexpected characters' (HTTP 400)
19	entrada inválida: patient_status	685848	"EN PROCESO"	TESTCOVER18	'Invalid status' (HTTP 422)
20	entrada inválida: patient_status	685848	12345	TESTCOVER19	'Invalid status: must be a string' (HTTP 400)
21	entrada inválida: patient_status	685848	12.345	TESTCOVER20	'Invalid status: must be a string' (HTTP 400)
22	entrada inválida: patient_status	685848	"	TESTCOVER21	'Invalid status: missing status' (HTTP 400)
23	entrada válida: id y patient_status	184750651	"PENDING"	TESTCOVER24, TESTCOVER33	lista de vinculaciones pendientes (HTTP 200)
24	entrada válida: id y patient_status	685848	"ACCEPTED"	TESTCOVER25, TESTCOVER28	lista vacía (HTTP 200)

Tabla 4.2: Casos de prueba derivados de los ítems de cobertura del endpoint "GET /vinculations/id"

Procedimiento de prueba (TD4)

Previo a la implementación de los casos de prueba, se identificaron datos de prueba faltantes para la ejecución de los casos de prueba. Específicamente para el caso de prueba 24, se requiere de un profesional de la salud que no cuente con vinculaciones. Se creó un usuario con rol de profesional de la salud y sin vinculaciones en la base de datos de prueba para estos casos.

En B.9 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento de prueba está desarrollado en Robot Framework, utilizando la biblioteca `RequestsLibrary` para realizar las solicitudes HTTP al endpoint. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

4.6. Segunda etapa: Gestión del entorno y datos de prueba

Establecimiento del entorno de prueba (ED1)

Se configuró un entorno de pruebas basado en Docker, que integró la implementación de los procedimientos de prueba desarrollados en Robot Framework (véase B.1).

La estructura de las carpetas del entorno de pruebas se puede ver en la figura B.5.

Preparación de los Datos de Prueba (ED2)

Se preparó una base de datos específica para las ejecuciones, cuya estructura y contenido se derivaron de los datos utilizados durante la presentación de la aplicación en la Feria de Software. Adicionalmente, se generaron datos complementarios que resultaron necesarios para la correcta ejecución de todos los casos de prueba.

Mantenimiento del Entorno de Prueba (ED3)

Se documentó el procedimiento para el despliegue del entorno de pruebas mediante Docker, detallando la configuración de los contenedores, la red interna y las versiones de todas las herramientas empleadas. Se garantizó que el entorno de pruebas estuviera disponible y operativo para la ejecución de las pruebas.

Mantenimiento de los Datos de Prueba (ED4)

Se formalizó el proceso para el mantenimiento y la actualización de la base de datos de prueba. Dicho proceso incluyó la creación de nuevos datos, la eliminación de registros obsoletos y la restauración de la base de datos a su estado inicial. De esta manera, se aseguró la disponibilidad y consistencia de los datos para cada ciclo de pruebas.

4.7. Tercera etapa: Ejecución de Pruebas

Ejecución de Procedimientos y Comparación de Resultados (TE1 y TE2)

Se llevaron a cabo todos los procedimientos de prueba diseñados para los endpoints identificados en la sección 4.4. La ejecución se realizó utilizando Robot Framework sobre el entorno de pruebas basado en Docker, lo que aseguró que cada prueba se desarrollara en condiciones controladas y reproducibles. Al finalizar la ejecución, se compararon los resultados obtenidos con las salidas esperadas definidas en los casos de prueba.

En la figura B.6 se adjunta un extracto de la salida de la terminal durante la ejecución del conjunto de pruebas.

Registro de la Ejecución de Pruebas (TE3)

Se generó de forma automatizada un registro (*log*) de la ejecución de todas las pruebas en formatos HTML y XML. Dicho registro incluyó el detalle de cada prueba, estadísticas de ejecución y un resumen general. En la figura B.7 se ilustra un fragmento del registro generado.

4.8. Cuarta etapa: Reporte de Incidentes

Análisis de Resultados de Pruebas (IR1)

A partir de los resultados de las pruebas ejecutadas, se identificaron nuevos incidentes, los cuales fueron debidamente documentados en el reporte de incidentes de pruebas. Se contempla como trabajo futuro el

análisis y la resolución de cada uno de estos incidentes.

Creación del Reporte de Incidentes (IR2)

Se documentaron todos los incidentes detectados durante la ejecución de las pruebas de manera automática tras la finalización de los procedimientos en Robot Framework.

Se generó un reporte de incidentes en formatos HTML y XML, el cual contenía el detalle de las pruebas ejecutadas, estadísticas sobre los resultados (pruebas omitidas, fallidas y exitosas) y un resumen consolidado de la información. En la imagen B.8 se presenta el reporte generado en formato HTML.

Este reporte de incidentes de pruebas constituye un insumo fundamental para la toma de decisiones y acciones correctivas futuras.

Capítulo 5

Análisis Retrospectivo

5.1. Dificultades presentadas en el desarrollo de la Memoria

Durante el desarrollo de la memoria se identificaron diversas dificultades asociadas tanto al contexto educacional en que se encontraba el software objeto de estudio como a la aplicación de un proceso formal de pruebas. Las principales dificultades fueron las siguientes:

- **Modelado de la arquitectura**: La arquitectura del producto de software no estaba ni descrita ni representada en diagramas. Fue necesario elaborarla desde cero a partir del análisis del código fuente, de las historias de usuario y de la interacción directa con la aplicación. Adicionalmente, se debió determinar el nivel de detalle más adecuado para su presentación.
- **Conocimiento de estándares de pruebas**: Si bien en la carrera se abordaron estándares generales de ingeniería de software, no existía formación previa en estándares específicos de pruebas. Por ello, resultó indispensable investigar y aprender el estándar ISO 29119:2021 para aplicarlo de manera rigurosa.
- **Aplicación de un estándar**: El trabajo con el estándar ISO 29119:2021 representó un desafío, dado que nunca se había trabajado previamente con un estándar formal de este nivel. Fue necesario recurrir a la orientación de profesores para comprender su uso y ajustarse al nivel de detalle exigido, particularmente considerando que se trataba de un trabajo de título.
- **Implementación de pruebas**: A pesar de haber cursado la asignatura de pruebas de software, fue necesario retomar conceptos, técnicas, herramientas y frameworks de *testing*. Ello implicó un esfuerzo adicional de reaprendizaje antes de poder aplicar los conocimientos en la implementación práctica.
- **Tiempo requerido**: La aplicación del proceso definido por el estándar ISO 29119:2021 demandó más tiempo del previsto. Por ejemplo, ejecutar el proceso completo para un único endpoint de la API, considerado como unidad de prueba, requirió al menos cinco horas de trabajo. Esto evidencia que un proceso de este nivel resulta poco viable en un contexto educacional caracterizado por limitaciones de tiempo y recursos.

- **Contexto de la Feria de Software**: El software sometido a pruebas fue desarrollado en el marco de la Feria de Software, lo que introdujo dificultades adicionales:
  - *Lapso temporal*: Mientras la feria correspondió a los cursos de noveno y décimo semestre, el presente trabajo corresponde a una asignatura del undécimo semestre. Esta brecha de, al menos, cinco meses obligó a retomar documentos, código y decisiones del proyecto, además de depender de la memoria individual del alumno.
  - *Colaboración del equipo*: No siempre es posible contar con la participación de los integrantes del equipo una vez finalizada la feria. En este caso se obtuvo retroalimentación puntual, pero la incertidumbre respecto de la disponibilidad de los integrantes constituyó una dificultad a considerar.

5.2. Hechos esperados y preconcebidos

A continuación, se presentan los hechos previstos organizados en torno a tres aspectos principales. En cada caso se detalla el contexto en que se sitúa el hecho y la relevancia que posee para el desarrollo del presente trabajo.

Esfuerzo requerido para aprender y aplicar pruebas de software

Se anticipaba que, durante el desarrollo de la memoria, sería necesario un esfuerzo considerable para adquirir y aplicar conocimientos vinculados a las pruebas de software. El plan de aprendizaje diseñado para este propósito se expone en el diagrama 1.2.

En dicho plan se identifican distintas etapas de estudio y práctica, las cuales demandaron una cantidad significativa de horas con el fin de aplicar las pruebas de manera efectiva. Entre estas etapas, la más relevante correspondió al estudio de los estándares de pruebas de software. En particular, la aplicación del estándar ISO 29119:2021 constituyó un desafío central, ya que implicó encontrar un equilibrio entre lo establecido en la norma y lo que era factible implementar en el contexto específico de esta memoria.

Cabe señalar que, a lo largo de la carrera, se mencionan estándares ampliamente reconocidos en el ámbito de la ingeniería informática. No obstante, estos suelen abordarse de manera superficial, sin llegar a una aplicación sistemática. Por esta razón, se esperaba que fuera necesario invertir un esfuerzo adicional en el aprendizaje y aplicación del estándar ISO 29119:2021, lo que efectivamente se confirmó durante la elaboración de esta memoria.

Formación previa en pruebas de software

El autor de esta memoria había cursado previamente la asignatura de pruebas de software, lo que generaba la expectativa de que el tiempo de dedicación requerido para aprender y aplicar técnicas de verificación y validación sería inferior al de un estudiante que no hubiera tenido dicha formación.

Sin embargo, durante el desarrollo del trabajo fue necesario destinar tiempo adicional a repasar conceptos ya conocidos, así como a adquirir nuevos enfoques y profundizar en aspectos que no habían sido

tratados en el curso. En consecuencia, la formación previa otorgó una ventaja relativa, pero no eximió de la necesidad de un proceso de aprendizaje complementario.

Además, debe considerarse que la asignatura de pruebas de software corresponde a un curso electivo dentro de la carrera, por lo que la mayoría de los estudiantes no posee conocimientos sólidos en esta materia. En consecuencia, el tiempo requerido no solo debe destinarse a la aplicación de las pruebas, sino también al aprendizaje de los conceptos y técnicas necesarias, lo que reduce aún más la viabilidad de realizar un proceso de pruebas exhaustivo.

Carga académica en contexto educacional como Feria de Software

En el desarrollo de la memoria fue posible concentrar de manera exclusiva los esfuerzos en la investigación y en la aplicación de pruebas de software, dado que el autor no contaba con asignaturas pendientes, prácticas profesionales ni otras responsabilidades académicas adicionales. Esta situación permitió disponer de tiempo suficiente para seguir el plan de aprendizaje propuesto y llevar a cabo su implementación en forma adecuada.

En contraste, en un contexto educacional como la Feria de Software, los estudiantes deben compatibilizar múltiples asignaturas y actividades paralelas, lo que restringe el tiempo disponible para la adquisición y aplicación de conocimientos sobre pruebas de software. Por ello, se esperaba que la dedicación efectiva a estas actividades en dicho contexto fuera significativamente menor a la observada en el desarrollo de esta memoria.

5.3. Hechos observados no previstos

A continuación, se presentan los hechos observados durante el desarrollo de la memoria que no fueron previstos desde el inicio y a través de la aplicación formal de un proceso de pruebas se pudieron identificar.

Adquisición de conocimiento del dominio a través de las pruebas de software

Se constató que la ejecución sistemática del proceso de pruebas de software durante el desarrollo de la memoria facilita una profundización progresiva en el **conocimiento del dominio** de la aplicación.

- **Contexto:** Este fenómeno se sitúa en la fase de aseguramiento de la calidad del software. La **ausencia de un cliente activo** en el proyecto, que funcionaría como fuente primaria de información del dominio, subraya la importancia de mecanismos internos de aprendizaje. La aplicación de técnicas de pruebas, particularmente las de caja negra (*black-box testing*), suple esta deficiencia.
- **Relevancia:** La relevancia es **altamente positiva**, ya que el incremento en el conocimiento del dominio revierte directamente en el beneficio del desarrollo del *software*. Esta información es crítica para la toma de decisiones informadas durante la fase de implementación y para asegurar la adecuación funcional del producto.

Un aspecto adicional a considerar es la utilidad del ***conformance testing*** (pruebas de conformidad) para la inmersión en el dominio. Al definir los casos de prueba, el enfoque se centra en el **comportamiento esperado** del sistema, disociado de su implementación interna. Esto obliga al analista a razonar sobre la lógica

inherente al dominio y sobre cómo el sistema debe interactuar dentro de dicho contexto, lo que resulta en una comprensión más sólida de sus requisitos funcionales.

Estos hechos se agrupan en torno a dos aspectos principales, detallando en cada caso el contexto en que se manifestaron y su relevancia para el trabajo realizado.

- El seguir rigurosamente un proceso de pruebas de software obliga (especialmente en el diseño de las pruebas) a generar una mejor caracterización del dominio (espacio del problema).
- La premura en la entrega de productos de trabajo requeridos en el contexto educacional de la feria de software, contraviene la interacción con el cliente, lo que dificulta la adecuada caracterización del dominio con la consecuente afectación de la calidad de los requisitos levantados.

5.4. Recomendaciones futuras para desarrollos en contextos educacionales

Incorporar pruebas de software en la formación académica

Resulta fundamental que los futuros ingenieros informáticos integren, desde su formación académica, la práctica sistemática de realizar pruebas sobre los desarrollos que llevan a cabo. Estas instancias de validación no deben limitarse exclusivamente a proyectos de gran envergadura, sino que también pueden aplicarse en programas simples, scripts o aplicaciones web de carácter básico.

La incorporación de pruebas de software constituye un elemento esencial para evidenciar que las soluciones desarrolladas cumplen con los objetivos previstos y que lo hacen de manera adecuada y confiable.

Para alcanzar este propósito, se considera necesario fomentar el desarrollo de habilidades de prueba en los estudiantes de forma paralela a sus competencias de programación. Asimismo, es recomendable promover una cultura académica en la cual se incentive a los estudiantes a verificar sus desarrollos bajo metodologías formales y guiadas, en lugar de recurrir a evaluaciones empíricas o informales.

Diseñar y documentar arquitectura del producto de software

La solicitud y elaboración de la arquitectura del producto de software constituye un aspecto esencial, ya que provee la base necesaria para fortalecer los atributos de calidad del software y ejecutar procesos de verificación y validación de manera adecuada.

La importancia de disponer de documentación arquitectónica se hace particularmente evidente al momento de formalizar las pruebas de software, en especial cuando se pretende aplicar un estándar reconocido como la norma ISO 29119:2021. No obstante, en proyectos desarrollados en un contexto educacional, la atención suele centrarse en la implementación de funcionalidades, relegando la generación de documentación de arquitectura, que en la mayoría de los casos no se elabora.

En consecuencia, resulta imperativo que en proyectos académicos se exija la definición y documentación de una arquitectura clara y estructurada, apoyada en herramientas formales como Visual Paradigm. De este modo, será posible no solo ejecutar pruebas con un nivel de formalidad adecuado, sino también evaluar de manera integral tanto la funcionalidad como los atributos de calidad del sistema.

Dar énfasis adicional a la priorización

En el ámbito profesional, la priorización de requisitos, pruebas de software y casos de uso constituye una práctica esencial para optimizar los recursos disponibles y cumplir de manera oportuna con los plazos establecidos.

En un contexto educacional, esta práctica adquiere aún mayor relevancia, dado que los proyectos se desarrollan dentro del mismo período académico que la asignatura, lo cual elimina cualquier margen de flexibilidad en la planificación. Por esta razón, resulta indispensable otorgar un énfasis adicional a la priorización, entendida como una competencia crítica que los estudiantes deben adquirir para desenvolverse de manera adecuada en su futuro desempeño profesional.

La priorización debe estar presente en todas las etapas del ciclo de vida del software: desde la definición de los requerimientos hasta la implementación y la ejecución de pruebas.

Dadas las restricciones temporales propias de un entorno académico, se vuelve fundamental que los estudiantes aprendan a establecer prioridades y a optimizar tanto su tiempo como sus recursos, con el fin de cumplir con los objetivos planteados y concretar las entregas dentro de los plazos estipulados.

Pruebas de regresión

Las pruebas de regresión constituyen una herramienta fundamental para la detección temprana de errores en el software, particularmente en lo que respecta a la implementación de historias de usuario.

En un contexto de desarrollo basado en metodologías ágiles, como el que se aplica en la Feria de Software, cada sprint incorpora modificaciones y nuevas funcionalidades sobre componentes ya existentes. Esta dinámica introduce un grado de incertidumbre respecto de la estabilidad del sistema, dado que los cambios recientes pueden afectar de manera imprevista el correcto funcionamiento de las partes previamente desarrolladas.

Tales inconsistencias no siempre resultan evidentes a simple vista, por lo que la ejecución sistemática de pruebas de regresión adquiere especial relevancia. Al aplicarlas, es posible identificar oportunamente los defectos introducidos, mitigar su impacto y garantizar una mayor confiabilidad en el producto final.

Como recomendación para futuros desarrollos en contextos educacionales, se sugiere aplicar pruebas de regresión de manera más flexible, es decir, sin necesidad de seguir de forma estricta un estándar de pruebas en toda su extensión. En su lugar, se plantea adoptar aquellos niveles y prácticas del estándar que resulten más pertinentes según las características del software en desarrollo. De este modo, se logra un equilibrio entre rigurosidad metodológica y viabilidad práctica, permitiendo mejorar la calidad del producto dentro de las limitaciones de tiempo y recursos propias de un entorno académico.

Capítulo 6

Conclusiones

El trabajo realizado en esta memoria evidencia la aplicación de un proceso formal de pruebas de software a una aplicación desarrollada en un contexto educacional, específicamente la aplicación ÚneteApp. Con el fin de evidenciar los hechos esperados y los hechos no previstos que emergen al aplicar pruebas de software en un contexto educacional, se realizó un análisis retrospectivo de los resultados obtenidos durante la aplicación del proceso de pruebas.

En cuanto a los objetivos específicos planteados, se logró:

- Definir y diseñar casos de prueba específicos para la aplicación ÚneteApp, basándonos en el estándar ISO 29119-2:2021, siguiendo el proceso de pruebas dinámicas a través de técnicas basadas en especificación sobre *endpoints* que fueron previamente identificados y priorizados.

- Aplicar pruebas de software para evaluar la aplicación ÚneteApp, lo que resultó en la identificación de errores y oportunidades de mejora, contribuyendo así a la identificación de aspectos de mejora para aumentar la calidad del software. Se dejaron documentados los procedimientos de prueba, la documentación e instrucciones necesarias para una futura ejecución y aplicación de las pruebas, así como los informes y reportes de prueba, identificando claramente qué puntos están fallando.

- Analizar hechos esperados y preconcebidos que emergen al aplicar pruebas de software a la aplicación ÚneteApp, lo que proporcionó una comprensión más profunda de los desafíos y beneficios asociados con las pruebas en un entorno educativo. Específicamente, se identificaron tres hechos esperados:

    - Esfuerzo requerido para aprender y aplicar pruebas de software.
    - Formación previa en pruebas de software.
    - Carga académica en contexto educacional.

- Analizar hechos observados no previstos que emergen al aplicar pruebas de software, lo que permitió reflexionar sobre el proceso y su impacto en el desarrollo del software en un contexto educacional. Principalmente, se identificaron dos hechos no previstos:

    - Conocimiento del dominio ganado a través de la aplicación rigurosa de pruebas de software.

- Impacto en la calidad del software debido a la inadecuada caracterización del dominio del software.

El alcance del trabajo se vio condicionado por las restricciones de tiempo, lo que obligó a priorizar las áreas del software sometidas a prueba, enfocándose en *ÚneteApp Health System*, donde se decidió concentrar los esfuerzos en cinco casos de uso específicos y en todos los *endpoints* derivados de ellos. Asimismo, fue necesario seleccionar las técnicas de diseño de pruebas más adecuadas, dado que aplicar la totalidad de ellas resultaba inviable en el marco temporal disponible. En este contexto, se optó por el uso de técnicas basadas en especificación para la elaboración de los casos de prueba, lo que permitió un diseño sistemático y consistente de las pruebas aplicadas.

Como trabajo futuro, se identifican diversas líneas de acción que pueden ser abordadas. Entre ellas, resulta pertinente la aplicación de otras técnicas de diseño de pruebas, así como la extensión del proceso a diferentes niveles de prueba. En esta memoria se llevaron a cabo principalmente pruebas unitarias; sin embargo, sería valioso incorporar pruebas de integración, de sistema y de aceptación, con el fin de obtener una visión más amplia de la calidad del software. Asimismo, se propone profundizar en la mejora de los aspectos observados durante la ejecución de las pruebas actuales, de manera que se contribuya a un incremento sostenido en la calidad de *ÚneteApp*.

Apéndice A

Diseño e implementación de las pruebas: Endpoints restantes

A.1. `PUT /vinculations/{id}`

Base de prueba

Se considera como ítem de prueba el endpoint "`PUT /vinculations/{id}`". A partir del ítem de prueba se deriva la base de prueba:

El endpoint recibe un parámetro de entrada `id` y cuerpo de la petición. El parámetro de ruta `id` corresponde al identificador asociado a la vinculación entre el profesional de la salud y el paciente a la cual se quiere modificar el estado. El cuerpo de la petición se compone solamente de `status` la que corresponde al nuevo estado de la vinculación entre el profesional de la salud y el paciente. Los valores permitidos para `status` son: "ACCEPTED", "REJECTED" y "PENDING".

La salida esperada corresponde a la vinculación actualizada, con el nuevo valor de estado.

Si se detecta una entrada inválida, la salida esperada es un mensaje de error en conjunto del código de estado HTTP correspondiente.

Se utilizará la técnica de particiones de equivalencia en conjunto con valores de frontera para derivar los ítems de cobertura.

El formato de una vinculación actualizada se puede ver en B.3.

Cobertura requerida del ítem de prueba

Cobertura completa de las particiones de equivalencia y valores de frontera identificados.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.1 que representa las particiones de equivalencia para el endpoint.

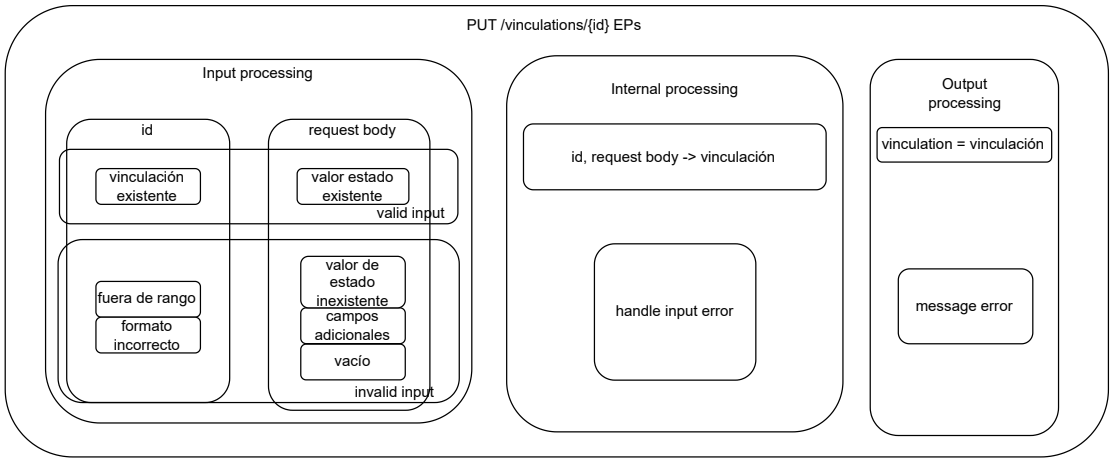


Figura A.1: Particiones de equivalencia del endpoint “PUT /vinculations/{id}”

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.1. Del procesamiento de la entrada:

- TESTCOVER1: id de una vinculación existente (1 <= id <= 38).
- TESTCOVER2: id de una vinculación inexistente (id <= 0).
- TESTCOVER3: id de una vinculación inexistente (39 <= id).
- TESTCOVER4: id con formato incorrecto (decimal).
- TESTCOVER5: id con formato incorrecto (string).
- TESTCOVER6: id con formato incorrecto (caracteres especiales).
- TESTCOVER7: id con formato incorrecto (booleano).
- TESTCOVER8: request body con estado válido.
- TESTCOVER9: request body con estado inexistente.
- TESTCOVER10: request body vacío.
- TESTCOVER11: request body con campos adicionales.

De los ítems de cobertura 1, 2 y 3 se identifican valores de frontera:

- BOUND1-inf: id = 1
- BOUND1-inf-out: id = 0
- BOUND1-sup: id = 38

- BOUND1-sup-out: id = 39
- BOUND2: id = 0
- BOUND2-out: id = 1
- BOUND3: id = 39
- BOUND3-out: id = 38

Del procesamiento interno:

- TESTCOVER12: salida con vinculación actualizada es inducida.
- TESTCOVER13: salida con mensaje de error es inducida.

Del procesamiento de la salida:

- TESTCOVER14: salida como vinculación actualizada y código HTTP 200.
- TESTCOVER15: salida como mensaje de error y código HTTP 4xx.

Derivar los casos de prueba (TD3)

Ahora habiendo identificado los ítems de cobertura, se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia y valores de forntera mínimos, es decir, cada caso de prueba cubre la mayor cantidad de ítems de cobertura posibles.

Se muestra en la tabla A.1 los casos de prueba derivados de las particiones de equivalencia y valores de frontera del endpoint. Todos los ítems de cobertura identificados en TD2 estan presentes en los casos de prueba, logrando un 100 % de cobertura.

Caso de prueba	EP involucrada	Entrada (id)	Entrada (request body)	Ítem de cobertura	Salida esperada
1	id y request body válido	1	{status: "REJECTED"}	TESTCOVER1, BOUND1-inf, TESTCOVER2, BOUND2-out, TESTCOVER8	vinculación actualizada (HTTP 200)
2	id inválido	0	{status: "PENDING"}	TESTCOVER1, BOUND1-inf-out, TESTCOVER2, BOUND2, TESTCOVER13, TESTCOVER15	'Vinculation not found'(HTTP 404)

Caso de prueba	EP involucrada	Entrada (id)	Entrada (request body)	Ítem de cobertura	Salida esperada
3	id y request body válido	38	{status: "ACCEPTED"}	TESTCOVER1, BOUND1-sup, TESTCOVER3, BOUND3-out, TESTCOVER12, TESTCOVER14	vinculación actualizada (HTTP 200)
4	id inválido	39	{status: "REJECTED"}	TESTCOVER1, BOUND1-sup-out, TESTCOVER3, BOUND3	'Vinculation not found' (HTTP 404)
5	id inválido	20.5	{status: "ACCEPTED"}	TESTCOVER4	'Invalid vinculation id' (HTTP 400)
6	id inválido	"20"	{status: "REJECTED"}	TESTCOVER5	'Invalid vinculation id' (HTTP 400)
7	id inválido	'20#'	{status: "PENDING"}	TESTCOVER6	'Invalid vinculation id' (HTTP 400)
8	id inválido	true	{status: "ACCEPTED"}	TESTCOVER7	'Invalid vinculation id' (HTTP 400)
9	request body inválido	20	{status: "COMPLETED"}	TESTCOVER1, TESTCOVER9	'Invalid status value' (HTTP 422)
10	request body inválido	20	{}	TESTCOVER1, TESTCOVER10	'Status is required' (HTTP 400)
11	request body inválido	20	{status: "PENDING", extra_field: 'extra'}	TESTCOVER1, TESTCOVER11	'Extra fields are not allowed' (HTTP 422)

Tabla A.1: Casos de prueba derivados de los ítems de cobertura del endpoint "PUT /vinculations/{id}"

Procedimiento de prueba (TD4)

En B.10 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

A.2. `POST /appointments`

Base de prueba

Se considera como ítem de prueba el endpoint "`POST /appointments`". A partir del ítem de prueba se deriva la base de prueba:

El endpoint recibe únicamente el cuerpo de la petición. Este se compone de los siguientes campos: `health_professional_id`, que corresponde al identificador del profesional de la salud y puede ser el RUT o el número de registro médico; `patient_id`, identificador del paciente (RUT); `reason`, motivo de consulta del paciente; `notes`, consideraciones registradas por el profesional de la salud; `dsm5`, estándar de clasificación de trastornos mentales; `cie11`, clasificación internacional de enfermedades; `eeag`, escala de evaluación de la actividad global; y `date`, que corresponde a la fecha en que ocurrió la consulta.

La salida esperada corresponde al registro de la consulta creada donde cada campo contiene la información entregada en el cuerpo de la petición.

Si se detecta una entrada inválida, la salida esperada es un mensaje de error en conjunto del código de estado HTTP correspondiente.

Se utilizará la técnica de particiones de equivalencia en conjunto con valores de frontera para derivar los ítems de cobertura, adicionalmente para las entradas de los campos `health_professional_id`, `patient_id` grafo causa-efecto.

El formato correspondiente a una consulta creada se puede ver en B.4.

Cobertura requerida del ítem de prueba

Cobertura completa de las particiones de equivalencia y valores de frontera identificados.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.2 que representa las particiones de equivalencia para el endpoint.

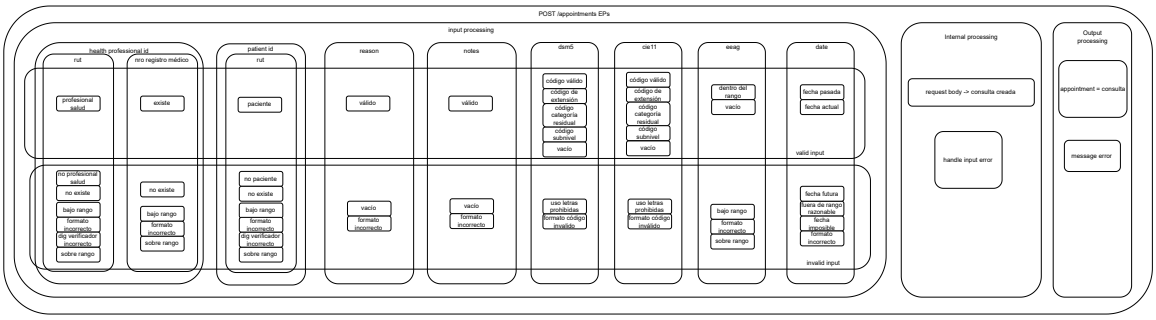


Figura A.2: Particiones de equivalencia del endpoint POST /appointments

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.2.
Del procesamiento de la entrada:

- TESTCOVER1: health professional id como rut correspondiente a un profesional de la salud (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: health professional id como número de registro médico existente (10 ≤ nro. registro médico ≤ 999.999).
- TESTCOVER3: health professional id como rut no correspondiente a un profesional de la salud.
- TESTCOVER4: health professional id como rut no existente.
- TESTCOVER5: health professional id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER6: health professional id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).

- TESTCOVER7: health professional id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER8: health professional id como rut con formato incorrecto (letras).
- TESTCOVER9: health professional id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER10: health professional id como rut con dígito verificador incorrecto.
- TESTCOVER11: health professional id como número de registro médico no existente.
- TESTCOVER12: health professional id como número de registro médico bajo el rango de registros médicos válidos (id ≤ 9).
- TESTCOVER13: health professional id como número de registro médico sobre el rango de registros médicos válidos (1.000.000 ≤ id).
- TESTCOVER14: health professional id como número de registro médico con formato incorrecto (caracteres especiales).
- TESTCOVER15: health professional id como número de registro médico con formato incorrecto (letras).
- TESTCOVER16: patient id como rut correspondiente a un paciente (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER17: patient id como rut no correspondiente a un paciente.
- TESTCOVER18: patient id como rut no existente.
- TESTCOVER19: patient id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER20: patient id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER21: patient id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER22: patient id como rut con formato incorrecto (letras).
- TESTCOVER23: patient id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER24: patient id como rut con dígito verificador incorrecto.
- TESTCOVER25: reason con formato correcto (string).
- TESTCOVER26: reason vacio.
- TESTCOVER27: reason con formato incorrecto (número).
- TESTCOVER28: reason con formato incorrecto (float).
- TESTCOVER29: notes con formato correcto (string).
- TESTCOVER30: notes vacio.
- TESTCOVER31: notes con formato incorrecto (número).

- TESTCOVER32: notes con formato incorrecto (float).
- TESTCOVER33: dsm5 como código válido (CIE-9-MC (CIE-10-MC)) [1].
- TESTCOVER34: dsm5 como código de extensión (prefijo X).
- TESTCOVER35: dsm5 como código con categoría residual (Y, Z, F, G terminales).
- TESTCOVER36: dsm5 como código de subniveles (con guion).
- TESTCOVER37: dsm5 vacío.
- TESTCOVER38: dsm5 con uso de letras prohibidas (I, O).
- TESTCOVER39: dsm5 con formato incorrecto (primer carácter incorrecto).
- TESTCOVER40: dsm5 con formato incorrecto (segundo carácter no es letra).
- TESTCOVER41: dsm5 con formato incorrecto (tercer carácter no es número).
- TESTCOVER42: dsm5 con formato incorrecto (estructura incompleta).
- TESTCOVER43: dsm5 con formato incorrecto (caracteres no permitidos).
- TESTCOVER44: cie11 como código válido (ED1E.EE) [2].
- TESTCOVER45: cie11 como código de extensión (prefijo X).
- TESTCOVER46: cie11 como código con categoría residual (Y, Z, F, G terminales).
- TESTCOVER47: cie11 como código de subniveles (con guion).
- TESTCOVER48: cie11 vacío.
- TESTCOVER49: cie11 con uso de letras prohibidas (I, O).
- TESTCOVER50: cie11 con formato incorrecto (primer carácter incorrecto).
- TESTCOVER51: cie11 con formato incorrecto (segundo carácter no es letra).
- TESTCOVER52: cie11 con formato incorrecto (tercer carácter no es número).
- TESTCOVER53: cie11 con formato incorrecto (estructura incompleta).
- TESTCOVER54: cie11 con formato incorrecto (caracteres no permitidos).
- TESTCOVER55: eeag dentro del rango válido ($1 \leq eeag \leq 100$) [3].
- TESTCOVER56: eeag vacío.
- TESTCOVER57: eeag bajo el rango válido ($eeag \leq 0$).

[1] Más información en [1]
[2] Más información en [26]
[3] Más información en [4]

- TESTCOVER58: eeag sobre el rango válido (101 ≤ eeag).
- TESTCOVER59: eeag con formato incorrecto (string).
- TESTCOVER60: eeag con formato incorrecto (float).
- TESTCOVER61: date con formato correcto (ISO 8601 YYYY-MM-DDTHH:MM:SS.mmmZ).
- TESTCOVER62: date como fecha pasada.
- TESTCOVER63: date como fecha actual.
- TESTCOVER64: date como fecha futura.
- TESTCOVER65: date como fecha fuera del rango razonable.
- TESTCOVER66: date como fecha imposible (ej 30 de febrero).
- TESTCOVER67: date con formato incorrecto (DD-MM-YYYY).
- TESTCOVER68: date con formato incorrecto (MM-DD-YYYY).
- TESTCOVER69: date con formato incorrecto (YYYY/MM/DD).
- TESTCOVER70: date con formato incorrecto (YYYYMMDD).
- TESTCOVER71: date con formato incorrecto (string no relacionado a fecha).
- TESTCOVER72: date con formato incorrecto (caracteres especiales).

De los ítems de cobertura 1, 2, 16 y 55 se identifican los siguientes valores de frontera:

- BOUND1-inf: health professional id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND1-inf-out: health professional id como rut fuera del límite inferior (rut = 999.999).
- BOUND1-sup: health professional id como rut dentro del límite superior (rut = 99.999.999).
- BOUND1-sup-out: health professional id como rut fuera del límite superior (rut = 100.000.000).
- BOUND2-inf: health professional id como número de registro médico dentro del límite inferior (nro. registro médico = 10).
- BOUND2-inf-out: health professional id como número de registro médico fuera del límite inferior (nro. registro médico= 9).
- BOUND2-sup: health professional id como número de registro médico dentro del límite superior (nro. registro médico = 999.999).
- BOUND2-sup-out: health professional id como número de registro médico fuera del límite superior (nro. registro médico = 1.000.000).
- BOUND16-inf: patient id como rut dentro del límite inferior (rut = 1.000.000).

- BOUND16-inf-out: patient id como rut fuera del límite inferior (rut = 999.999).
- BOUND16-sup: patient id como rut dentro del límite superior (rut = 99.999.999).
- BOUND16-sup-out: patient id como rut fuera del límite superior (rut = 100.000.000).
- BOUND55-inf: eeag dentro del límite inferior (eeag = 1).
- BOUND55-inf-out: eeag fuera del límite inferior (eeag = 0).
- BOUND55-sup: eeag dentro del límite superior (eeag = 100).
- BOUND55-sup-out: eeag fuera del límite superior (eeag = 101).

Del procesamiento interno:

- TESTCOVER73: salida con consulta creada es inducida.
- TESTCOVER74: salida con mensaje de error es inducida.

Del procesamiento de la salida:

- TESTCOVER75: salida como consulta creada y código HTTP 201.
- TESTCOVER76: salida como mensaje de error y código HTTP 4xx.

Ítems de cobertura adicionales por grafo causa-efecto en las entradas `health_professional_id` y `patient_id`:

Se tienen las siguientes causas:

- C1: health professional id corresponde a un profesional de la salud.
- C2: health professional id correspondea un paciente.
- C3: patient id corresponde a un paciente.
- C4: patient id corresponde a un profesional de la salud.

Se tienen los siguientes efectos:

- E1: Respuesta con consulta creada (HTTP 201).
- E2: Respuesta con mensaje de error (HTTP 404).

Y las siguientes relaciones:

- TESTCOVER77: C1 `AND` C3 → E1
- TESTCOVER78: C1 `AND` C4 → E2
- TESTCOVER79: C2 `AND` C3 → E2
- TESTCOVER80: C2 `AND` C4 → E2

Derivar los casos de prueba (TD3)

Ahora habiendo identificado los ítems de cobertura, se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia y valores de forntera mínimos, es decir, cada caso de prueba cubre la mayor cantidad de ítems de cobertura posibles.

Se muestra en la tabla A.2 los casos de prueba derivados de las particiones de equivalencia y valores de frontera del endpoint. Se logran cubrir todos los ítems de cobertura identificados en TD2.

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
1-1	health professional id como rut válido	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER1, TESTCOVER73, TESTCOVER75	consulta creada (HTTP 201)
1-2	health professional id como rut inválido	{health_professional_id: '1.000.000-9', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER1, TESTCOVER4, TESTCOVER74, TESTCOVER76, BOUND1-inf	'Health professional not found' (HTTP 404)
1-3	health professional id como rut inválido	{health_professional_id: '999.999-K', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER1, TESTCOVER5, BOUND1-inf-out	'Invalid id: length violation' (HTTP 422)
1-4	health professional id como rut inválido	{health_professional_id: '99.999.999-9', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER1, TESTCOVER4, BOUND1-sup	'Health professional not found' (HTTP 404)
1-5	health professional id como rut inválido	{health_professional_id: '100.000.000-7', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER1, TESTCOVER6, BOUND1-sup-out	'Invalid id: length violation' (HTTP 422)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
2-1	health professional id como número registro médico válido	{health_professional_id: 550129, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER2	consulta creada (HTTP 201)
2-2	health professional id como número registro médico inválido	{health_professional_id: 10, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER2, TESTCOVER11, BOUND2-inf	'Health professional not found' (HTTP 404)
2-3	health professional id como número registro médico inválido	{health_professional_id: 9, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER2, TESTCOVER12, BOUND2-inf-out	'Invalid id: length violation' (HTTP 422)
2-4	health professional id como número registro médico inválido	{health_professional_id: 999999, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER2, TESTCOVER11, BOUND2-sup	'Health professional not found' (HTTP 404)
2-5	health professional id como número registro médico inválido	{health_professional_id: 1000000, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER2, TESTCOVER13, BOUND2-sup-out	'Invalid id: length violation' (HTTP 422)
3	health professional id como rut inválido	{health_professional_id: '22.208.530-6', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER3	'Health professional not found' (HTTP 404)
4	health professional id como rut inválido	{health_professional_id: '22.208.530-6', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER4	'Health professional not found' (HTTP 404)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
7	health professional id como rut inválido	{health_professional_id: '12.110.471*7', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER7	'Invalid id: unexpected characters' (HTTP 400)
8	health professional id como rut inválido	{health_professional_id: '12A110471-7', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER8	'Invalid id: unexpected characters' (HTTP 400)
9	health professional id como rut inválido	{health_professional_id: '18.475.065', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER9	'Invalid id: missing check digit' (HTTP 400)
10	health professional id como rut inválido	{health_professional_id: '18.475.065-2', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER10	'Invalid id: check digit' (HTTP 400)
11	health professional id como número de registro médico no existente	{health_professional_id: 123457, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER11	'Health professional not found' (HTTP 404)
12	health professional id como número de registro médico bajo el rango válido	{health_professional_id: 9, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER12	'Invalid id: length violation' (HTTP 422)
13	health professional id como número de registro médico sobre el rango válido	{health_professional_id: 1000000, patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER13	'Invalid id: length violation' (HTTP 422)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
14	health professional id como número de registro médico con caracteres especiales	{health_professional_id: '55@0129', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER14	'Invalid id: unexpected characters' (HTTP 400)
15	health professional id como número de registro médico con letras	{health_professional_id: 'A50129', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER15	'Invalid id: unexpected characters' (HTTP 400)
16	patient id como rut correspondiente a un paciente	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER16	consulta creada (HTTP 201)
17	patient id como rut no correspondiente a un paciente	{health_professional_id: '18.475.065-1', patient_id: '18.475.065-1', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER17	'Patient not found' (HTTP 404)
18	patient id como rut no existente	{health_professional_id: '18.475.065-1', patient_id: '21.222.333-6', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER18	'Patient not found' (HTTP 404)
19	patient id como rut bajo el rango válido	{health_professional_id: '18.475.065-1', patient_id: '999.999-9', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER19	'Invalid id: length violation' (HTTP 422)
20	patient id como rut sobre el rango válido	{health_professional_id: '18.475.065-1', patient_id: '100.000.000-7', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER20	'Invalid id: length violation' (HTTP 422)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
21	patient id con formato incorrecto (caracteres especiales)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173*0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER21	'Invalid id: unexpected characters' (HTTP 400)
22	patient id con formato incorrecto (letras)	{health_professional_id: '18.475.065-1', patient_id: '20A957173-0', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER22	'Invalid id: unexpected characters' (HTTP 400)
23	patient id sin dígito verificador	{health_professional_id: '18.475.065-1', patient_id: '20.957.173', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER23	'Invalid id: missing check digit' (HTTP 400)
24	patient id con dígito verificador incorrecto	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-9', reason: 'Malestar general y sensación de ansiedad', notes: 'Paciente llega decaido', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER24	'Invalid id: check digit' (HTTP 400)
25	reason con formato correcto (string)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Dolor torácico leve', notes: 'Sin antecedentes relevantes', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER25	consulta creada (HTTP 201)
26	reason vacío	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: '', notes: 'Sin antecedentes relevantes', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER26	'Invalid reason: required' (HTTP 400)
27	reason con formato incorrecto (número)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 123, notes: 'Sin antecedentes relevantes', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER27	'Invalid reason: type' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
28	reason con formato incorrecto (float)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 1.23, notes: 'Sin antecedentes relevantes', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER28	'Invalid reason: type' (HTTP 400)
29	notes con formato correcto (string)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Dolor torácico leve', notes: 'Paciente consciente, estable', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER29	consulta creada (HTTP 201)
30	notes vacío	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Dolor torácico leve', notes: '', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER30	'Invalid notes: required' (HTTP 400)
31	notes con formato incorrecto (número)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Dolor torácico leve', notes: 123, dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER31	'Invalid notes: type' (HTTP 400)
32	notes con formato incorrecto (float)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Dolor torácico leve', notes: 1.23, dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER32	'Invalid notes: type' (HTTP 400)
33	dsm5 válido	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER33	consulta creada (HTTP 201)
34	dsm5 código de extensión (prefijo X)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'X32.0', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER34	consulta creada (HTTP 201)
35	dsm5 categoría residual (terminal Y/Z/F/G)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F99', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER35	consulta creada (HTTP 201)
36	dsm5 con guion (subniveles)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F32-0', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER36	consulta creada (HTTP 201)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
37	dsm5 vacío	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER37	consulta creada (HTTP 201)
38	dsm5 con letras prohibidas (I/O)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'I10', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER38	'Invalid dsm5: forbidden letters' (HTTP 422)
39	dsm5 primer carácter incorrecto	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '132.10', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER39	'Invalid dsm5: bad first character' (HTTP 400)
40	dsm5 segundo carácter no es letra	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F@2.0', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER40	'Invalid dsm5: bad second character' (HTTP 400)
41	dsm5 tercer carácter no es número	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F3A.0', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER41	'Invalid dsm5: bad third character' (HTTP 400)
42	dsm5 estructura incompleta	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F3', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER42	'Invalid dsm5: incomplete' (HTTP 400)
43	dsm5 con caracteres no permitidos	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: 'F32.0#', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER43	'Invalid dsm5: unexpected characters' (HTTP 400)
44	cie11 válido	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'ED1E.EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER44	consulta creada (HTTP 201)
45	cie11 código de extensión (prefijo X)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'XK8M.2', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER45	consulta creada (HTTP 201)
46	cie11 categoría residual (terminal Y/Z/F/G)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'ZZ9Z.ZZ', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER46	consulta creada (HTTP 201)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
47	cie11 con guion (subniveles)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'ED1E-EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER47	consulta creada (HTTP 201)
48	cie11 vacío	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: '', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER48	consulta creada (HTTP 201)
49	cie11 con letras prohibidas (I/O)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'OI1E.EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER49	'Invalid cie11: forbidden letters' (HTTP 422)
50	cie11 primer carácter incorrecto	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: '1D1E.EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER50	'Invalid cie11: bad first character' (HTTP 400)
51	cie11 segundo carácter no es letra	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'E11E.EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER51	'Invalid cie11: bad second character' (HTTP 400)
52	cie11 tercer carácter no es número	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'EDAE.EE', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER52	'Invalid cie11: bad third character' (HTTP 400)
53	cie11 estructura incompleta	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'ED1E', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER53	'Invalid cie11: incomplete' (HTTP 400)
54	cie11 con caracteres no permitidos	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'ED1E.EE#', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER54	'Invalid cie11: unexpected characters' (HTTP 400)
55	eeag dentro del rango válido (1–100)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER55	consulta creada (HTTP 201)
56	eeag vacío	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: '', date: '2025-09-03T14:54:03.871Z'}	TESTCOVER56	consulta creada (HTTP 201)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
57	eeag bajo el rango válido (≤ 0)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 0, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER57	'Invalid eeag: out of range' (HTTP 422)
58	eeag sobre el rango válido (≥ 101)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 101, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER58	'Invalid eeag: out of range' (HTTP 422)
59	eeag con formato incorrecto (string)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 'setenta y ocho', date: '2025-09-03T14:54:03.871Z'}	TESTCOVER59	'Invalid eeag: type' (HTTP 400)
60	eeag con formato incorrecto (float)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78.5, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER60	'Invalid eeag: type' (HTTP 400)
61	date con formato correcto (ISO 8601)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871Z'}	TESTCOVER61	consulta creada (HTTP 201)
62	date como fecha pasada	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2024-05-01T10:00:00.000Z'}	TESTCOVER62	consulta creada (HTTP 201)
63	date como fecha actual	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-04T00:00:00.000Z'}	TESTCOVER63	consulta creada (HTTP 201)
64	date como fecha futura	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-12-01T10:00:00.000Z'}	TESTCOVER64	'Invalid date: future appointments are not allowed' (HTTP 422)
65	date fuera de rango razonable	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '1900-01-01T00:00:00.000Z'}	TESTCOVER65	'Invalid date: out of reasonable range' (HTTP 422)
66	date imposible	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-02-30T10:00:00.000Z'}	TESTCOVER66	'Invalid date: impossible date' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
67	date con formato incorrecto (DD-MM-YYYY)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '03-09-2025'}	TESTCOVER67	'Invalid date: format' (HTTP 400)
68	date con formato incorrecto (MM-DD-YYYY)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '09-03-2025'}	TESTCOVER68	'Invalid date: format' (HTTP 400)
69	date con formato incorrecto (YYYY/MM/DD)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025/09/03'}	TESTCOVER69	'Invalid date: format' (HTTP 400)
70	date con formato incorrecto (YYYYMMDD)	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '20250903'}	TESTCOVER70	'Invalid date: format' (HTTP 400)
71	date con string no relacionado	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: 'ayer a las dos'}	TESTCOVER71	'Invalid date: format' (HTTP 400)
72	date con caracteres especiales	{health_professional_id: '18.475.065-1', patient_id: '20.957.173-0', reason: 'Ansiedad', notes: 'Control', dsm5: '296.21 (F32.0)', cie11: 'MG25', eeag: 78, date: '2025-09-03T14:54:03.871!'}	TESTCOVER72	'Invalid date: unexpected characters' (HTTP 400)
73	health professional id es un profesional y patient id es un paciente	{"health_professional_id": "18.475.065-1", patient_id: "20.957.173-0", reason: "Malestar general", notes: "Paciente llega decaido", dsm5: "296.21 (F32.0)", cie11: "MG25", eeag: 78, date: "2025-09-04T10:00:00.000Z"}	TESTCOVER77	consulta creada (HTTP 201)
74	health professional id es un profesional y patient id es un profesional	{health_professional_id: "18.475.065-1", patient_id: "18.475.065-1", reason: "Malestar general", notes: "Paciente llega decaido", dsm5: "296.21 (F32.0)", cie11: "MG25", eeag: 78, date: "2025-09-04T10:00:00.000Z"}	TESTCOVER78	'Patient not found' (HTTP 404)
75	health professional id es un paciente y patient id es un paciente	{health_professional_id: "20.957.173-0", patient_id: "20.957.173-0", reason: "Malestar general", notes: "Paciente llega decaido", dsm5: "296.21 (F32.0)", cie11: "MG25", eeag: 78, date: "2025-09-04T10:00:00.000Z"}	TESTCOVER79	'Health professional not found' (HTTP 404)

Caso de prueba	EP involucrada	Entrada (request body)	Ítem de cobertura	Salida esperada
76	health professional id es un paciente y patient id es un profesional	{health_professional_id: "20.957.173-0", patient_id: "18.475.065-1", reason: "Malestar general", notes: "Paciente llega decaido", dsm5: "296.21 (F32.0)", cie11: "MG25", eeag: 78, date: "2025-09-04T10:00:00.000Z"}	TESTCOVER80	'Health professional not found' (HTTP 404)

Tabla A.2: Casos de prueba derivados de los ítems de cobertura del endpoint “POST /appointments”

Procedimiento de prueba (TD4)

En B.11 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

A.3. `GET /appointments/{health_professional_id}`

Base de prueba

Endpoint `GET /appointments/health_professional_id` considerado como ítem de prueba. A partir de este ítem se deriva la base de prueba:

El endpoint recibe dos parámetros de entrada: `health_professional_id` y `patient_id`. El primero es un parámetro de ruta obligatorio y corresponde al identificador del profesional de la salud. El segundo es un parámetro de consulta opcional correspondiente al identificador del paciente.

Se reconocen como identificadores válidos de profesionales de la salud tanto los RUTs como los números de registro médico. Para los pacientes, se consideran RUTs.

La salida esperada corresponde a una lista que contiene el historial de consultas médicas del profesional de la salud. En caso de que se especifique un `patient_id`, la salida esperada es el historial de consultas del profesional asociadas exclusivamente a dicho paciente. Si no existen consultas registradas, la respuesta corresponde a una lista vacía.

En caso de que se detecte una entrada inválida, la salida esperada corresponde a un mensaje de error junto con el código de estado HTTP asociado.

Para el la derivación de los ítems de cobertura se emplearán las técnicas particiones de equivalencia en conjunto con valores frontera y grafo causa-efecto.

Cada elemento de la lista de consultas sigue el formato mostrado en B.5.

Cobertura requerida del ítem de prueba

Cobertura completa de los ítems de prueba derivados de las clases de equivalencia, valores de frontera y grafo causa-efecto.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.3 que representa las particiones de equivalencia para el endpoint.

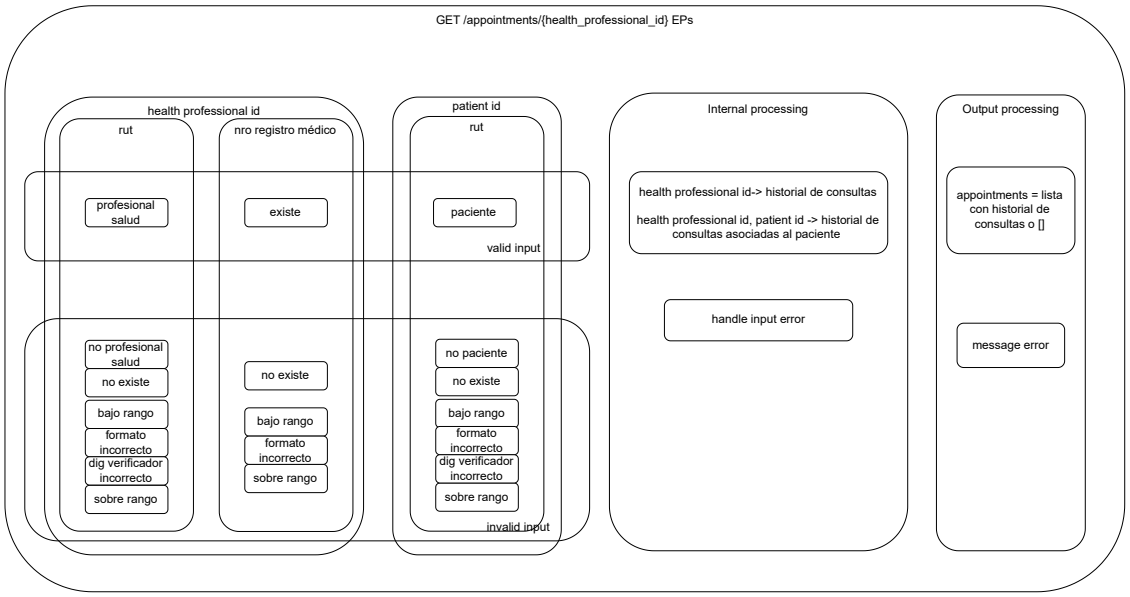


Figura A.3: Particiones de equivalencia del endpoint "GET /appointments/{health_professional_id}"

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.3. Del procesamiento de la entrada:

- TESTCOVER1: health professional id como rut correspondiente a un profesional de la salud (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: health professional id como número de registro médico existente (10 ≤ nro. registro médico ≤ 999.999).
- TESTCOVER3: health professional id como rut no correspondiente a un profesional de la salud.
- TESTCOVER4: health professional id como rut no existente.
- TESTCOVER5: health professional id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER6: health professional id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER7: health professional id como rut con formato incorrecto (caracteres especiales).

- TESTCOVER8: health professional id como rut con formato incorrecto (letras).
- TESTCOVER9: health professional id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER10: health professional id como rut con dígito verificador incorrecto.
- TESTCOVER11: health professional id como número de registro médico no existente.
- TESTCOVER12: health professional id como número de registro médico bajo el rango de registros médicos válidos (id ≤ 9).
- TESTCOVER13: health professional id como número de registro médico sobre el rango de registros médicos válidos (1.000.000 ≤ id).
- TESTCOVER14: health professional id como número de registro médico con formato incorrecto (caracteres especiales).
- TESTCOVER15: health professional id como número de registro médico con formato incorrecto (letras).
- TESTCOVER16: patient id como rut correspondiente a un paciente (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER17: patient id como rut no correspondiente a un paciente.
- TESTCOVER18: patient id como rut no existente.
- TESTCOVER19: patient id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER20: patient id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER21: patient id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER22: patient id como rut con formato incorrecto (letras).
- TESTCOVER23: patient id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER24: patient id como rut con dígito verificador incorrecto.

De los ítems de cobertura 1, 2 y 16 se identifican los siguientes valores de frontera:

- BOUND1-inf: health professional id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND1-inf-out: health professional id como rut fuera del límite inferior (rut = 999.999).
- BOUND1-sup: health professional id como rut dentro del límite superior (rut = 99.999.999).
- BOUND1-sup-out: health professional id como rut fuera del límite superior (rut = 100.000.000).
- BOUND2-inf: health professional id como número de registro médico dentro del límite inferior (nro. registro médico = 10).
- BOUND2-inf-out: health professional id como número de registro médico fuera del límite inferior (nro. registro médico= 9).

- BOUND2-sup: health professional id como número de registro médico dentro del límite superior (nro. registro médico = 999.999).
- BOUND2-sup-out: health professional id como número de registro médico fuera del límite superior (nro. registro médico = 1.000.000).
- BOUND16-inf: patient id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND16-inf-out: patient id como rut fuera del límite inferior (rut = 999.999).
- BOUND16-sup: patient id como rut dentro del límite superior (rut = 99.999.999).
- BOUND16-sup-out: patient id como rut fuera del límite superior (rut = 100.000.000).

Del procesamiento interno:

- TESTCOVER25: lista con historial de consultas de un profesional de la salud es inducido.
- TESTCOVER26: lista con historial de consultas de un profesional de la salud asociado a un paciente es inducido.
- TESTCOVER27: mensaje de error es inducido.

Del procesamiento de la salida:

- TESTCOVER28: salida con historial de consultas y código HTTP 200.
- TESTCOVER29: salida con lista vacía de historial de consultas y código HTTP 200.
- TESTCOVER32: salida con mensaje de error y código HTTP 4xx.

Ítems de cobertura adicionales por grafo causa-efecto en las entradas `health_professional_id` y `patient_id`:

Se tienen las siguientes causas:

- C1: health professional id corresponde a un profesional de la salud.
- C2: health professional id corresponde a un paciente.
- C3: patient id corresponde a un paciente.
- C4: patient id corresponde a un profesional de la salud.

Se tienen los siguientes efectos:

- E1: Respuesta con historial de consultas (HTTP 200).
- E2: Respuesta con mensaje de error (HTTP 404).

Y las siguientes relaciones:

- TESTCOVER33: C1 AND C3 → E1
- TESTCOVER34: C1 AND C4 → E2
- TESTCOVER35: C2 AND C3 → E2
- TESTCOVER36: C2 AND C4 → E2

Derivar los casos de prueba (TD3)

Se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia uno a uno, es decir, cada caso de prueba cubre solo un ítem de cobertura.

Se muestra en la tabla A.3 los casos de prueba derivados de las particiones de equivalencia del endpoint.

Caso de prueba	EP involucrada	Entrada (health_professional_id)	Entrada (patient_id)	Ítem de cobertura	Salida esperada
1	valid input	18.475.065-1	20.957.173-0	TESTCOVER1, TESTCOVER16, TESTCOVER26, TESTCOVER28, TESTCOVER33	lista de consultas (HTTP 200)
2	invalid input: health professional id	1.000.000-9	20.957.173-0	TESTCOVER4, BOUND1-inf, TESTCOVER27, TESTCOVER32	'Health professional not found' (HTTP 404)
3	invalid input: health professional id	999.999-K	20.957.173-0	TESTCOVER5, BOUND1-inf-out	'Invalid id: length violation' (HTTP 422)
4	invalid input: health professional id	99.999.999-9	20.957.173-0	TESTCOVER4, BOUND1-sup	'Health professional not found' (HTTP 404)
5	invalid input: health professional id	100.000.000-7	20.957.173-0	TESTCOVER6, BOUND1-sup-out	'Invalid id: length violation' (HTTP 422)
6	valid input	550129	20.957.173-0	TESTCOVER2, TESTCOVER16	lista de consultas (HTTP 200)
7	invalid input: health professional id	10	20.957.173-0	TESTCOVER11, BOUND2-inf	'Health professional not found' (HTTP 404)
8	invalid input: health professional id	9	20.957.173-0	TESTCOVER12, BOUND2-inf-out	'Invalid id: length violation' (HTTP 422)
9	invalid input: health professional id	999999	20.957.173-0	TESTCOVER11, BOUND2-sup	'Health professional not found' (HTTP 404)
10	invalid input: health professional id	1000000	20.957.173-0	TESTCOVER13, BOUND2-sup-out	'Invalid id: length violation' (HTTP 422)
11	invalid input: health professional id	22.208.530-6	20.957.173-0	TESTCOVER3, TESTCOVER4	'Health professional not found' (HTTP 404)
12	invalid input: health professional id	12.110.471*7	20.957.173-0	TESTCOVER7	'Invalid id: unexpected characters' (HTTP 400)
13	invalid input: health professional id	12A110471-7	20.957.173-0	TESTCOVER8	'Invalid id: unexpected characters' (HTTP 400)
14	invalid input: health professional id	18.475.065	20.957.173-0	TESTCOVER9	'Invalid id: missing check digit' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (health_professional_id)	Entrada (patient_id)	Ítem de cobertura	Salida esperada
15	invalid input: health professional id	18.475.065-2	20.957.173-0	TESTCOVER10	'Invalid id: check digit' (HTTP 400)
16	invalid input: health professional id	55@0129	20.957.173-0	TESTCOVER14	'Invalid id: unexpected characters' (HTTP 400)
17	invalid input: health professional id	A50129	20.957.173-0	TESTCOVER15	'Invalid id: unexpected characters' (HTTP 400)
18	invalid input: patient id	18.475.065-1	18.475.065-1	TESTCOVER17, TESTCOVER34	'Patient not found' (HTTP 404)
19	invalid input: patient id	18.475.065-1	21.222.333-6	TESTCOVER18	'Patient not found' (HTTP 404)
20	invalid input: patient id	18.475.065-1	999.999-9	TESTCOVER19, BOUND16-inf-out	'Invalid id: length violation' (HTTP 422)
21	invalid input: patient id	18.475.065-1	100.000.000-7	TESTCOVER20, BOUND16-sup-out	'Invalid id: length violation' (HTTP 422)
22	invalid input: patient id	18.475.065-1	20.957.173*0	TESTCOVER21	'Invalid id: unexpected characters' (HTTP 400)
23	invalid input: patient id	18.475.065-1	20A957173-0	TESTCOVER22	'Invalid id: unexpected characters' (HTTP 400)
24	invalid input: patient id	18.475.065-1	20.957.173	TESTCOVER23	'Invalid id: missing check digit' (HTTP 400)
25	invalid input: patient id	18.475.065-1	20.957.173-9	TESTCOVER24	'Invalid id: check digit' (HTTP 400)
26	valid input	18.999.590-3		TESTCOVER1, TESTCOVER25, TESTCOVER29	lista vacía de consultas (HTTP 200)
27	invalid input: health professional id	20.957.173-0	20.957.173-0	TESTCOVER35	'Health professional not found' (HTTP 404)
28	invalid input: health professional id	20.957.173-0	18.475.065-1	TESTCOVER36	'Health professional not found' (HTTP 404)

Tabla A.3: Casos de prueba derivados de los ítems de cobertura del endpoint "GET /appointments/"

Fuente: Elaboración propia

Procedimiento de prueba (TD4)

En B.12 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

A.4. `GET /last_diagnosis`

Base de prueba

Se considera como ítem de prueba el endpoint `GET /last_diagnosis`. A partir de este ítem se deriva la base de prueba:

El endpoint recibe `health_professional_id` y `patient_id` como entrada. El primer parámetro de consulta corresponde al identificador del profesional de la salud y el segundo es opcional correspondiente al identificador del paciente. Ambos parámetros de consulta son obligatorios.

Se reconocen como identificadores válidos de profesionales de la salud tanto los RUTs como los números de registro médico. Para los pacientes, se consideran RUTs.

La salida esperada corresponde al último diagnóstico del profesional asociado a dicho paciente. Si no existen consultas registradas, la respuesta corresponde a un mensaje indicando que no existen diagnósticos registrados.

En caso de que se detecte una entrada inválida, la salida esperada corresponde a un mensaje de error junto con el código de estado HTTP asociado.

Para la derivación de los ítems de cobertura se emplearán las técnicas de particiones de equivalencia en conjunto con valores frontera y grafo causa-efecto.

El formato de la respuesta esperada correspondiente a un diagnóstico se muestra en B.6.

Cobertura requerida del ítem de prueba

Cobertura completa de los ítems de prueba derivados de las clases de equivalencia, valores de frontera y grafo causa-efecto.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.4 que representa las particiones de equivalencia para el endpoint.

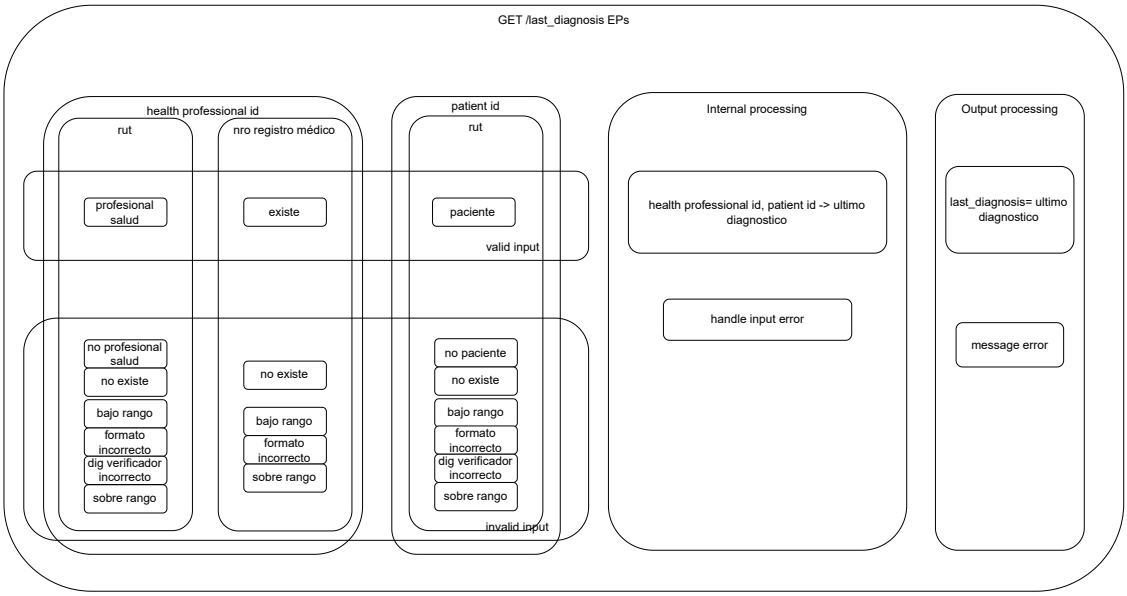


Figura A.4: Particiones de equivalencia del endpoint ”GET /last_diagnosis”

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.4. Del procesamiento de la entrada:

- TESTCOVER1: health professional id como rut correspondiente a un profesional de la salud (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: health professional id como número de registro médico existente (10 ≤ nro. registro médico ≤ 999.999).
- TESTCOVER3: health professional id como rut no correspondiente a un profesional de la salud.
- TESTCOVER4: health professional id como rut no existente.
- TESTCOVER5: health professional id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER6: health professional id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER7: health professional id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER8: health professional id como rut con formato incorrecto (letras).
- TESTCOVER9: health professional id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER10: health professional id como rut con dígito verificador incorrecto.
- TESTCOVER11: health professional id como número de registro médico no existente.

- TESTCOVER12: health professional id como número de registro médico bajo el rango de registros médicos válidos (id ≤ 9).
- TESTCOVER13: health professional id como número de registro médico sobre el rango de registros médicos válidos (1.000.000 ≤ id).
- TESTCOVER14: health professional id como número de registro médico con formato incorrecto (caracteres especiales).
- TESTCOVER15: health professional id como número de registro médico con formato incorrecto (letras).
- TESTCOVER16: patient id como rut correspondiente a un paciente (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER17: patient id como rut no correspondiente a un paciente.
- TESTCOVER18: patient id como rut no existente.
- TESTCOVER19: patient id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER20: patient id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER21: patient id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER22: patient id como rut con formato incorrecto (letras).
- TESTCOVER23: patient id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER24: patient id como rut con dígito verificador incorrecto.

De los ítems de cobertura 1, 2 y 16 se identifican los siguientes valores de frontera:

- BOUND1-inf: health professional id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND1-inf-out: health professional id como rut fuera del límite inferior (rut = 999.999).
- BOUND1-sup: health professional id como rut dentro del límite superior (rut = 99.999.999).
- BOUND1-sup-out: health professional id como rut fuera del límite superior (rut = 100.000.000).
- BOUND2-inf: health professional id como número de registro médico dentro del límite inferior (nro. registro médico = 10).
- BOUND2-inf-out: health professional id como número de registro médico fuera del límite inferior (nro. registro médico= 9).
- BOUND2-sup: health professional id como número de registro médico dentro del límite superior (nro. registro médico = 999.999).
- BOUND2-sup-out: health professional id como número de registro médico fuera del límite superior (nro. registro médico = 1.000.000).

- BOUND16-inf: patient id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND16-inf-out: patient id como rut fuera del límite inferior (rut = 999.999).
- BOUND16-sup: patient id como rut dentro del límite superior (rut = 99.999.999).
- BOUND16-sup-out: patient id como rut fuera del límite superior (rut = 100.000.000).

Del procesamiento interno:

- TESTCOVER25: último diagnóstico de un profesional de la salud asociado a un paciente es inducido.
- TESTCOVER26: mensaje de error es inducido.

Del procesamiento de la salida:

- TESTCOVER27: salida con último diagnóstico y código HTTP 200.
- TESTCOVER28: salida con mensaje indicando que no existen diagnósticos registrados y código HTTP 404.
- TESTCOVER29: salida con mensaje de error y código HTTP 4xx.

Ítems de cobertura adicionales asociado a la relación entre `health_professional_id` y `patient_id`: Se tienen las siguientes causas:

- C1: health professional tiene pacientes.
- C2: health professional no tiene pacientes.
- C3: patient asociado a profesional de la salud.
- C4: patient no asociado a profesinal de la salud.

Se tienen los siguientes efectos:

- E1: Respuesta con último diagnóstico (HTTP 200).
- E2: Respuesta con mensaje que no existen diagnósticos registrados (HTTP 404).

Y las siguientes relaciones:

- TESTCOVER30: C1 AND C3 → E1
- TESTCOVER31: C1 AND C3 → E2
- TESTCOVER32: C1 AND C4 → E2
- TESTCOVER33: C2 AND C3 → E2

Derivar los casos de prueba (TD3)

Se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia uno a uno, es decir, cada caso de prueba cubre solo un ítem de cobertura.

Se muestra en la tabla A.4 los casos de prueba derivados de las particiones de equivalencia del endpoint.

Caso de prueba	EP involucrada	Entrada (health_professional_id)	Entrada (patient_id)	Ítem de cobertura	Salida esperada
1	valid input	18.475.065-1	20.957.173-0	TESTCOVER1, TESTCOVER16, TESTCOVER25, TESTCOVER27	último diagnóstico (HTTP 200)
2	invalid input: health professional id	1.000.000-9	20.957.173-0	TESTCOVER4, TESTCOVER26, TESTCOVER29, BOUND1-inf,	'Health professional not found' (HTTP 404)
3	invalid input: health professional id	999.999-K	20.957.173-0	TESTCOVER5, BOUND1-inf-out	'Invalid id: length violation' (HTTP 422)
4	invalid input: health professional id	99.999.999-9	20.957.173-0	TESTCOVER4, BOUND1-sup	'Health professional not found' (HTTP 404)
5	invalid input: health professional id	100.000.000-7	20.957.173-0	TESTCOVER6, BOUND1-sup-out	'Invalid id: length violation' (HTTP 422)
6	valid input	550129	20.957.173-0	TESTCOVER2, TESTCOVER16	último diagnóstico (HTTP 200)
7	invalid input: health professional id	10	20.957.173-0	TESTCOVER11, BOUND2-inf	'Health professional not found' (HTTP 404)
8	invalid input: health professional id	9	20.957.173-0	TESTCOVER12, BOUND2-inf-out	'Invalid id: length violation' (HTTP 422)
9	invalid input: health professional id	999999	20.957.173-0	TESTCOVER11, BOUND2-sup	'Health professional not found' (HTTP 404)
10	invalid input: health professional id	1000000	20.957.173-0	TESTCOVER13, BOUND2-sup-out	'Invalid id: length violation' (HTTP 422)
11	invalid input: health professional id	22.208.530-6	20.957.173-0	TESTCOVER3, TESTCOVER4	'Health professional not found' (HTTP 404)
12	invalid input: health professional id	12.110.471*7	20.957.173-0	TESTCOVER7	'Invalid id: unexpected characters' (HTTP 400)
13	invalid input: health professional id	12A110471-7	20.957.173-0	TESTCOVER8	'Invalid id: unexpected characters' (HTTP 400)
14	invalid input: health professional id	18.475.065	20.957.173-0	TESTCOVER9	'Invalid id: missing check digit' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (health_professional_id)	Entrada (patient_id)	Ítem de cobertura	Salida esperada
15	invalid input: health professional id	18.475.065-2	20.957.173-0	TESTCOVER10	'Invalid id: check digit' (HTTP 400)
16	invalid input: health professional id	55@0129	20.957.173-0	TESTCOVER14	'Invalid id: unexpected characters' (HTTP 400)
17	invalid input: health professional id	A50129	20.957.173-0	TESTCOVER15	'Invalid id: unexpected characters' (HTTP 400)
18	invalid input: patient id	18.475.065-1	18.475.065-1	TESTCOVER17	'Patient not found' (HTTP 404)
19	invalid input: patient id	18.475.065-1	21.222.333-6	TESTCOVER18	'Patient not found' (HTTP 404)
20	invalid input: patient id	18.475.065-1	999.999-9	TESTCOVER19, BOUND16-inf-out	'Invalid id: length violation' (HTTP 422)
21	invalid input: patient id	18.475.065-1	100.000.000-7	TESTCOVER20, BOUND16-sup-out	'Invalid id: length violation' (HTTP 422)
22	invalid input: patient id	18.475.065-1	20.957.173*0	TESTCOVER21	'Invalid id: unexpected characters' (HTTP 400)
23	invalid input: patient id	18.475.065-1	20A957173-0	TESTCOVER22	'Invalid id: unexpected characters' (HTTP 400)
24	invalid input: patient id	18.475.065-1	20.957.173	TESTCOVER23	'Invalid id: missing check digit' (HTTP 400)
25	invalid input: patient id	18.475.065-1	20.957.173-9	TESTCOVER24	'Invalid id: check digit' (HTTP 400)
26	valid input	18.999.590-3	20.957.173-0	TESTCOVER1, TESTCOVER25, TESTCOVER28	'Last diagnosis not found' (HTTP 404)
27	valid input	18.475.065-1	20.957.173-0	TESTCOVER30	último diagnóstico (HTTP 200)
28	valid input	18.999.590-3	20.957.173-0	TESTCOVER31	'Last diagnosis not found' (HTTP 404)
29	valid input	18.475.065-1	21.222.333-6	TESTCOVER32	'Last diagnosis not found' (HTTP 404)
30	valid input	18.999.590-3	20.957.173-0	TESTCOVER33	'Last diagnosis not found' (HTTP 404)

Tabla A.4: Casos de prueba derivados de los ítems de cobertura del endpoint "GET /last_diagnosis"

Procedimiento de prueba (TD4)

En B.13 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

A.5. `GET /prescription/status/{patient_id}`

Base de prueba

A partir del ítem de prueba `GET /prescription/status/{patient_id}` se deriva la siguiente base de prueba:

El endpoint recibe un parámetro como entrada `patient_id`. Este corresponde a un parámetro de ruta obligatorio y corresponde al identificador del paciente. Se consideran como identificadores válidos de pacientes los RUTs.

La salida esperada corresponde al estado del paciente respecto a la prescripción social. En caso de que se detecte una entrada inválida, la salida esperada corresponde a un mensaje de error junto con el código de estado HTTP asociado.

Para el la derivación de los ítems de cobertura se emplearán las técnicas particiones de equivalencia en conjunto con valores frontera.

El formato de la salida asociado al estado de un paciente con respecto a la prescripción social se puede ver en B.7.

Cobertura requerida del ítem de prueba

Cobertura completa de los ítems de prueba identificados de las clases de equivalencia y valores de frontera.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.5 que representa las particiones de equivalencia para el endpoint.

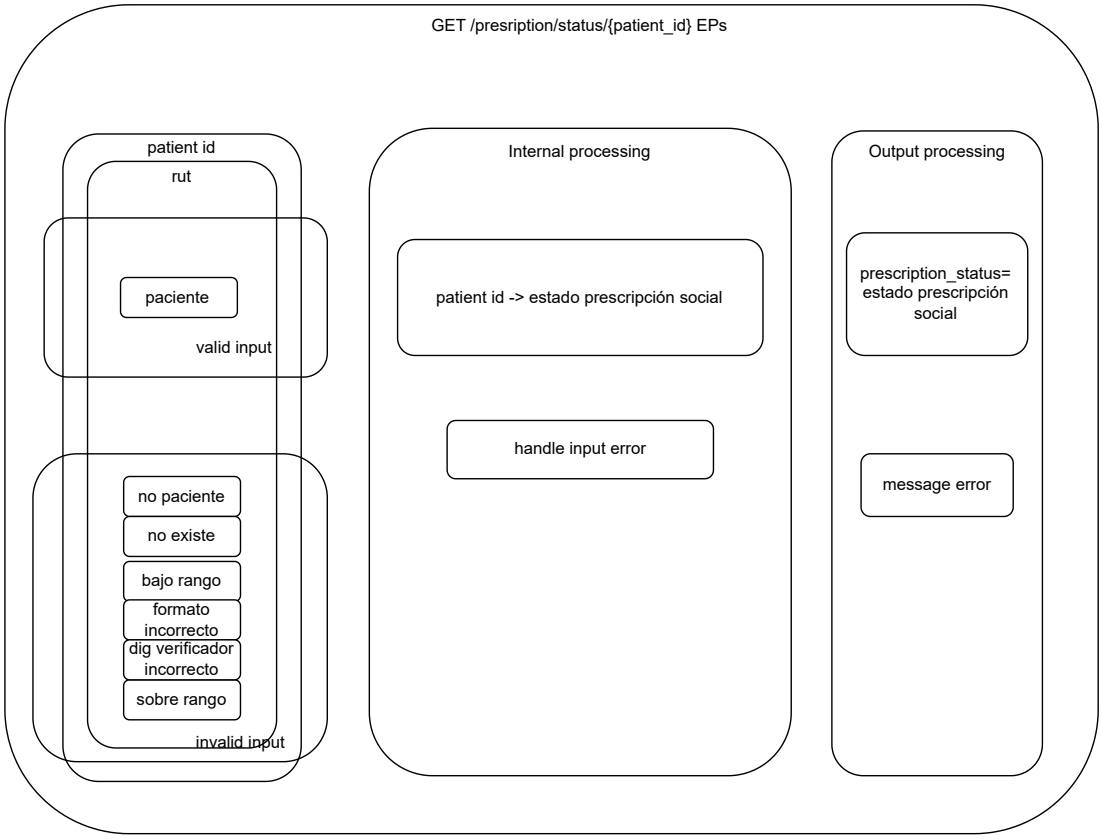


Figura A.5: Particiones de equivalencia del endpoint ”GET /prescription/status/{patient_id}”

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.5. Del procesamiento de la entrada:

- TESTCOVER1: patient id como rut correspondiente a un paciente (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: patient id como rut no correspondiente a un paciente.
- TESTCOVER3: patient id como rut no existente.
- TESTCOVER4: patient id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER5: patient id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER6: patient id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER7: patient id como rut con formato incorrecto (letras).
- TESTCOVER8: patient id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER9: patient id como rut con dígito verificador incorrecto.

Del ítem de cobertura 1 se identifican los siguientes valores de frontera:

- BOUND1-inf: patient id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND1-inf-out: patient id como rut fuera del límite inferior (rut = 999.999).
- BOUND1-sup: patient id como rut dentro del límite superior (rut = 99.999.999).
- BOUND1-sup-out: patient id como rut fuera del límite superior (rut = 100.000.000).

Del procesamiento interno:

- TESTCOVER10: estado de prescripción social de paciente es inducido.
- TESTCOVER11: mensaje de error es inducido.

Del procesamiento de la salida:

- TESTCOVER12: salida con estado de prescripción social y código HTTP 200.
- TESTCOVER13: salida con mensaje de error y código HTTP 4xx.

Derivar los casos de prueba (TD3)

Se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia uno a uno, es decir, cada caso de prueba cubre solo un ítem de cobertura.

Se muestra en la tabla A.5 los casos de prueba derivados de las particiones de equivalencia del endpoint.

Caso de prueba	EP involucrada	Entrada (patient_id)	Ítem de cobertura	Salida esperada
1	valid input	20.957.173-0	TESTCOVER1, TESTCOVER10, TESTCOVER12	estado de prescripción social de paciente (HTTP 200)
2	invalid input	18.475.065-1	TESTCOVER2, TESTCOVER11, TESTCOVER13	'Patient not found' (HTTP 404)
3	invalid input	21.222.333-6	TESTCOVER3	'Patient not found' (HTTP 404)
4	invalid input id	1.000.000-9	TESTCOVER4, BOUND1-inf	'Patient not found' (HTTP 404)
5	invalid input id	999.999-9	TESTCOVER4, BOUND1-inf-out	'Invalid id: length violation' (HTTP 422)
6	invalid input id	100.000.000-7	TESTCOVER5, BOUND1-sup-out	'Invalid id: length violation' (HTTP 422)
7	invalid input id	99.999.999-9	TESTCOVER5, BOUND1-sup	'Patient not found' (HTTP 404)
8	invalid input id	20.957.173*0	TESTCOVER6	'Invalid id: unexpected characters' (HTTP 400)
9	invalid input id	20A957173-0	TESTCOVER7	'Invalid id: unexpected characters' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (patient_id)	Ítem de cobertura	Salida esperada
10	invalid input id	20.957.173	TESTCOVER8	'Invalid id: missing check digit' (HTTP 400)
11	invalid input id	20.957.173-9	TESTCOVER9	'Invalid id: check digit' (HTTP 400)

Tabla A.5: Casos de prueba derivados de los ítems de cobertura del endpoint "GET /prescription/status/{patient_id}"

Procedimiento de prueba (TD4)

En B.14 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio úneteapp test.

A.6. `POST /prescription/{patient_id}`

Base de prueba

A partir del ítem de prueba `POST /prescription/{patient_id}` se deriva la siguiente base de prueba:

El endpoint recibe como entrada `patient_id`. Este corresponde a un parámetro de ruta obligatorio y corresponde al identificador del paciente. Se consideran como identificadores válidos de pacientes los RUTs.

La salida esperada corresponde a la actividad prescrita al paciente. En caso de que se detecte una entrada inválida, la salida esperada corresponde a un mensaje de error junto con el código de estado HTTP asociado.

Para el la derivación de los ítems de cobertura se emplearán las técnicas particiones de equivalencia en conjunto con valores frontera.

El formato de la salida asociado a la actividad obtenida de la prescripción social se puede ver en B.8.

Cobertura requerida del ítem de prueba

Cobertura completa de los ítems de prueba identificados de las clases de equivalencia y valores de frontera.

Modelo de pruebas (TD1)

Se desarrolla un diagrama A.6 que representa las particiones de equivalencia para el endpoint.

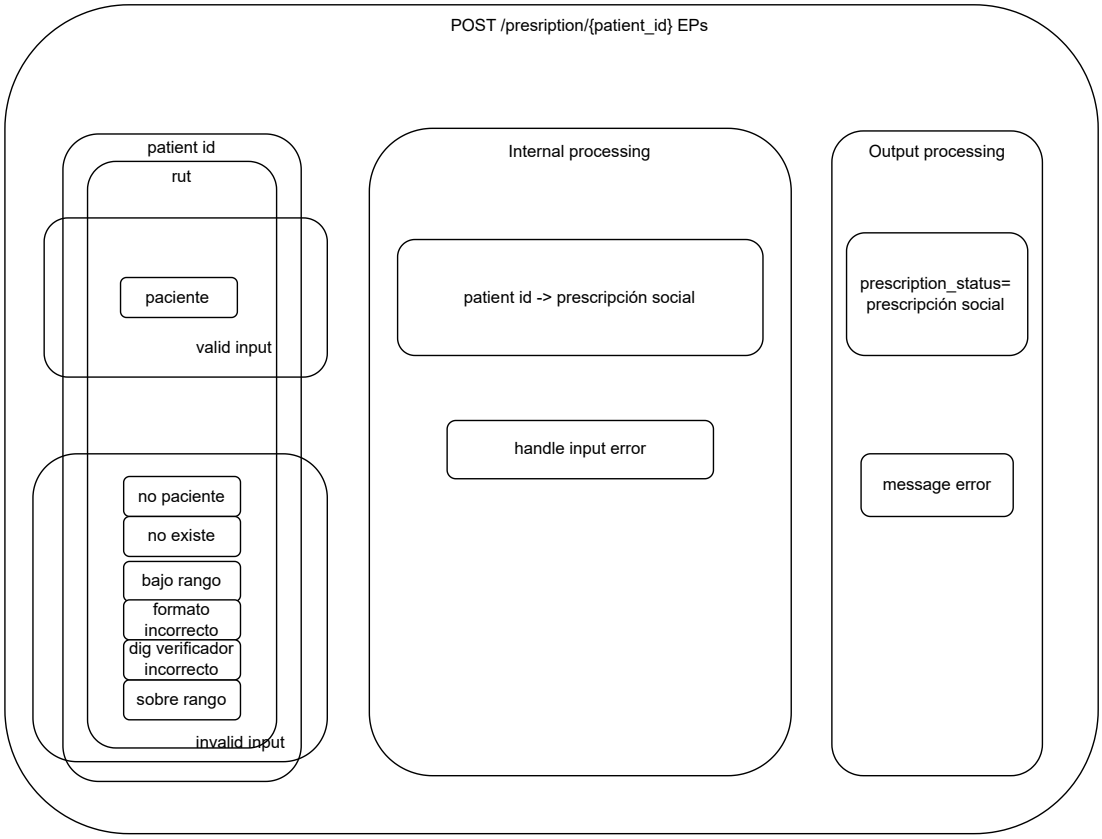


Figura A.6: Particiones de equivalencia del endpoint “POST /prescription/{patient_id}”

Ítems de cobertura de prueba (TD2)

Los ítems de cobertura de prueba son las particiones de equivalencia identificadas en el diagrama A.6. Del procesamiento de la entrada:

- TESTCOVER1: patient id como rut correspondiente a un paciente (1.000.000 ≤ rut ≤ 99.999.999).
- TESTCOVER2: patient id como rut no correspondiente a un paciente.
- TESTCOVER3: patient id como rut no existente.
- TESTCOVER4: patient id como rut bajo el rango de ruts válidos (rut ≤ 999.999).
- TESTCOVER5: patient id como rut sobre el rango de ruts válidos (100.000.000 ≤ rut).
- TESTCOVER6: patient id como rut con formato incorrecto (caracteres especiales).
- TESTCOVER7: patient id como rut con formato incorrecto (letras).
- TESTCOVER8: patient id como rut con formato incorrecto (sin dígito verificador).
- TESTCOVER9: patient id como rut con dígito verificador incorrecto.

Del ítem de cobertura 1 se identifican los siguientes valores de frontera:

- BOUND1-inf: patient id como rut dentro del límite inferior (rut = 1.000.000).
- BOUND1-inf-out: patient id como rut fuera del límite inferior (rut = 999.999).
- BOUND1-sup: patient id como rut dentro del límite superior (rut = 99.999.999).
- BOUND1-sup-out: patient id como rut fuera del límite superior (rut = 100.000.000).

Del procesamiento interno:

- TESTCOVER10: actividad derivada de la prescripción social de paciente es inducido.
- TESTCOVER11: mensaje de error es inducido.

Del procesamiento de la salida:

- TESTCOVER12: salida con prescripción social y código HTTP 201.
- TESTCOVER13: salida con mensaje de error y código HTTP 4xx.

Derivar los casos de prueba (TD3)

Se derivan casos de prueba intentando asegurar la cobertura completa de los ítems de cobertura. Se derivan los casos de prueba de las particiones de equivalencia mínimos, es decir, cada caso de prueba cubre la mayor cantidad de ítems de cobertura posibles.

Se muestra en la tabla A.6 los casos de prueba derivados de las particiones de equivalencia del endpoint.

Caso de prueba	EP involucrada	Entrada (patient_id)	Ítem de cobertura	Salida esperada
1	valid input	20.463.976-0	TESTCOVER1, TESTCOVER10, TESTCOVER12	prescripción social de paciente (HTTP 201)
2	invalid input	18.475.065-1	TESTCOVER2, TESTCOVER11, TESTCOVER13	'Patient not found' (HTTP 404)
3	invalid input	21.222.333-6	TESTCOVER3	'Patient not found' (HTTP 404)
4	invalid input id	1.000.000-9	TESTCOVER4, BOUND1-inf	'Patient not found' (HTTP 404)
5	invalid input id	999.999-9	TESTCOVER4, BOUND1-inf-out	'Invalid id: length violation' (HTTP 422)
6	invalid input id	100.000.000-7	TESTCOVER5, BOUND1-sup-out	'Invalid id: length violation' (HTTP 422)
7	invalid input id	99.999.999-9	TESTCOVER5, BOUND1-sup	'Patient not found' (HTTP 404)
8	invalid input id	20.463.976*0	TESTCOVER6	'Invalid id: unexpected characters' (HTTP 400)
9	invalid input id	20A957173-0	TESTCOVER7	'Invalid id: unexpected characters' (HTTP 400)

Caso de prueba	EP involucrada	Entrada (patient_id)	Ítem de cobertura	Salida esperada
10	invalid input id	20.463.976	TESTCOVER8	'Invalid id: missing check digit' (HTTP 400)
11	invalid input id	20.463.976-9	TESTCOVER9	'Invalid id: check digit' (HTTP 400)

Tabla A.6: Casos de prueba derivados de ítems de cobertura de endpoint “POST /prescription/{patient_id}”

Procedimiento de prueba (TD4)

En B.15 se muestra un fragmento del procedimiento de prueba desarrollado. En este procedimiento solamente se muestra el primer caso de prueba, pero el procedimiento completo incluye todos los casos de prueba derivados. El procedimiento incluye la configuración inicial (bibliotecas utilizadas), la definición de variables (entorno y url del endpoint), la ejecución de cada caso de prueba y la verificación de las salidas esperadas. El procedimiento completo se encuentra en el siguiente repositorio público enlace.

Apéndice B

Anexos

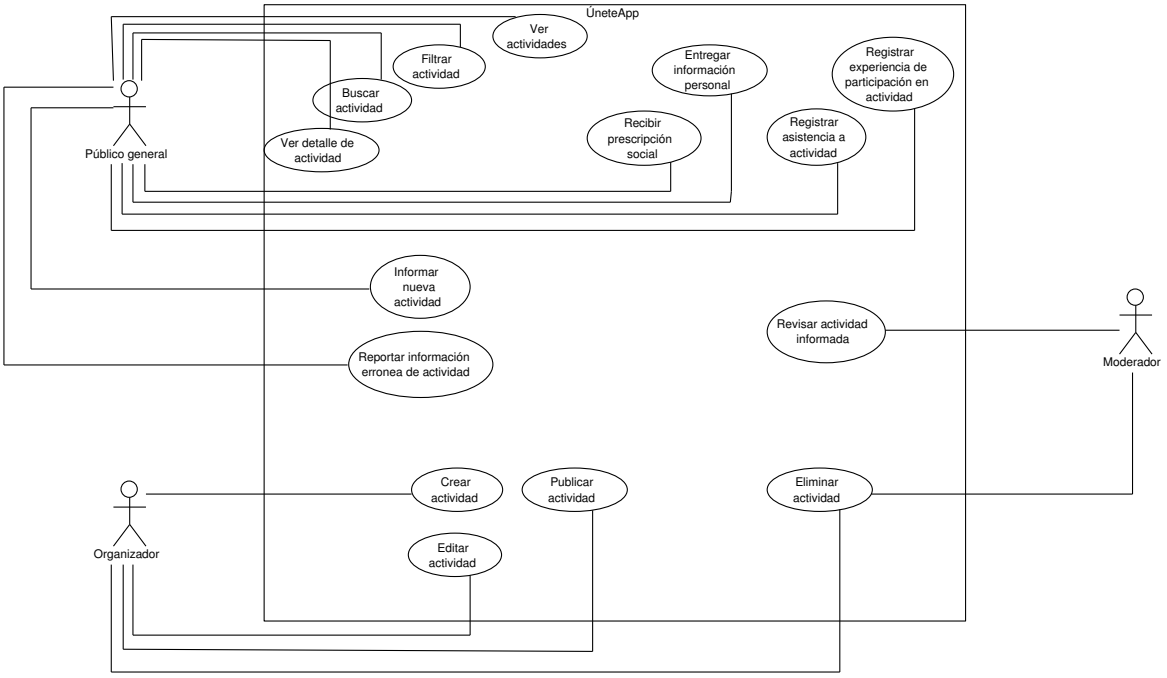


Figura B.1: Diagrama de casos de uso de ÚneteApp

Sprint 1		
Historia de usuario	**Puntos de historia**	**Categoría**
Spike-1	7	Crítico
Spike-2	10	Crítico
HU-4 Información personal	7	Crítico
HU-3 Recibir prescripción	7	Crítico
HU-6 Asistencia a actividad	8	Crítico
HU-5 Experiencia en actividad	14	Crítico
HU-28 Sección Mis actividades (prescritas y de interés)	8	Crítico
Total	61	

Figura B.2: Historias de usuario Sprint 1

Fuente: Equipo ÚneteApp, INF-360 e INF-228 2024

Sprint 2		
Historia de usuario	**Puntos de historia**	**Categoría**
HU-1 Ver actividades	2	Alta
HU-2 Ver detalle actividad	2	Alta
HU-8 Búsqueda actividades	5	Alta
HU-9 Filtro actividades	5	Alta
HU-10 Informar actividad no existente en app	6	Alta
HU-11 Reportar información errónea	4	Alta
HU-12 Publicar actividad	5	Alta
HU-13 Editar actividad	6	Alta
HU-14 Revisión de actividades informadas	6	Alta
HU-29 Mis pacientes	20	Alta
Total	61	

Figura B.3: Historias de usuario Sprint 2

Fuente: Equipo ÚneteApp, INF-360 e INF-228 2024

Sprint 3		
Historia de usuario	**Puntos de historia**	**Categoría**
HU-33 Eventos	3	Medio
HU-34 Crear Eventos Moderador/Organizador/Participante(Joven)	3	Medio
HU-35 Sección mis actividades/eventos organizador	5	Medio
HU-36 Personalización de monitoreo de experiencia	13	Medio
HU-37 Agregar preguntas de monitoreo	5	Medio
HU-38 Detalle de monitoreo	5	Medio
HU-39 Reportar problema en actividad	1/2	Medio
HU-40 Registrar consulta	3	Medio
Total	37,5	

Figura B.4: Historias de usuario Sprint 3

Fuente: Equipo ÚneteApp, INF-360 e INF-228 2024

```yaml
name: uneteapp-test
services:
  postgres-test:
    image: postgres:16
    restart: always
    volumes:
      - dbdata_test:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    environment:
      - DATABASE_HOST=127.0.0.1
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=uneteapp12345
      - POSTGRES_DB=repository
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres -d repository"]
      interval: 5s
      timeout: 5s
      retries: 10

  backend-test:
    build:
      context: ./services/repository
      dockerfile: Dockerfile.test
    ports:
      - "8000:8000"
    volumes:
      - ./services/repository/src:/app/src
    depends_on:
      postgres-test:
        condition: service_healthy

  test:
    build:
      context: ./tests
      dockerfile: Dockerfile.test
    environment:
      DB_HOST: postgres-test
      DB_PORT: "5432"
      DB_USER: postgres
      DB_PASSWORD: uneteapp12345
      DB_NAME: repository
      BACKUP_PATH: /backups/uneteapp_backup_14-11-2024_02_05_plain.sql
    depends_on:
```

```
      postgres-test:
        condition: service_healthy
    volumes:
      - ./tests:/workspace/tests
      - ./tests/backups:/backups:ro
    command: [ "sh", "-c", "./tests/reset-bd.sh && sleep infinity" ]

volumes:
  dbdata_test:
```

Listing B.1: Ejemplo de archivo `docker-compose.test.yml`

```
{
    "id": int,
    "patient_id": int,
    "patient_info": {
        "name": str,
        "lastname": str,
        "gender": str,
        "email": str,
        "phone": str
    },
    "created_at": datetime,
    "updated_at": datetime,
    "status": str
}
```

Listing B.2: Formato de elemento vinculaciones

```
{
  "patient_id": int,
  "doctor_id": int,
  "status": str,
  "created_at": datetime,
  "updated_at": datetime
}
```

Listing B.3: Formato de elemento vinculación actualizada

```
{
    "id": int,
    "health_professional_id": int,
    "patient_id": int,
    "reason": str,
    "notes": str,
    "dsm5": str,
```

```
    "cie11": str,
    "eeag": str,
    "date": datetime
}
```

Listing B.4: Formato de elemento consulta creada

```
{
    "id": int,
    "health_professional_id": int,
    "patient_id": int,
    "reason": str,
    "notes": str,
    "dsm5": str,
    "cie11": str,
    "eeag": str,
    "date": datetime
}
```

Listing B.5: Formato de un elemento tipo consulta

```
{
    "dsm5": str,
    "cie11": str,
    "eeag": str
}
```

Listing B.6: Formato de último diagnóstico.

```
{
  "patient_id": int,
  "ready_for_prescription": bool,
  "ready_to_answer": bool,
  "count_answered_required_questions": int,
  "days_since_last_prescription": int,
  "has_prescription": bool
}
```

Listing B.7: Formato salida estado prescripción social.

```
{
  "patient_id": int,
  "activity_id": int,
  "active": bool,
  "assists": int,
  "not_assists": int,
```

```
  "created_at": datetime,
  "updated_at": datetime,
  "latest_attendance_response": datetime,
  "is_prescription": bool,
  "id": int,
  "activity": {
      "name": str,
      "user_reporter_id": int,
      "description": str,
      "image_url": str,
      "address": str,
      "city": str,
      "active": bool,
      "activity_type": str,
      "organizer": str,
      "phone": str,
      "email": str,
      "website": str,
      "status": str,
      "participants": int,
      "validation_status": str,
      "participation_requirement": str,
      "created_at": datetime,
      "updated_at": datetime,
      "id": int,
      "schedules": [
      {
          "day": str,
          "start_time": str,
          "end_time": str,
          "active": bool,
          "created_at": datetime,
          "updated_at": datetime
      },
      ...
    ]
  }
}
```

Listing B.8: Formato actividad prescripción social.

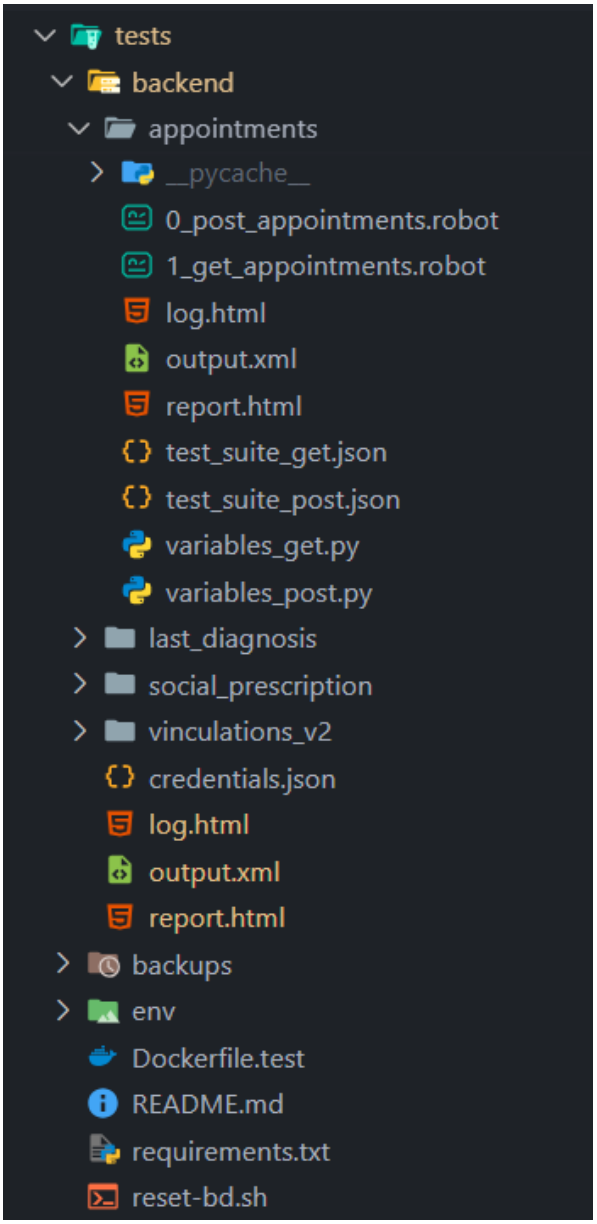


Figura B.5: Estructura de carpetas del entorno de pruebas

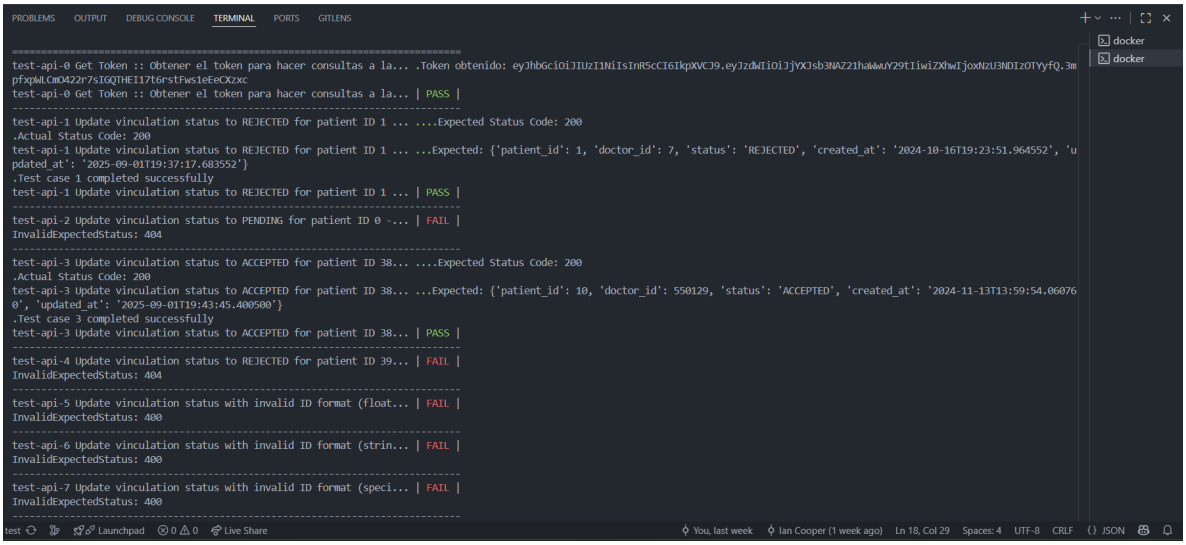


Figura B.6: Fragmento de la ejecución de pruebas

Figura B.7: Fragmento del log de ejecución generado por Robot Framework

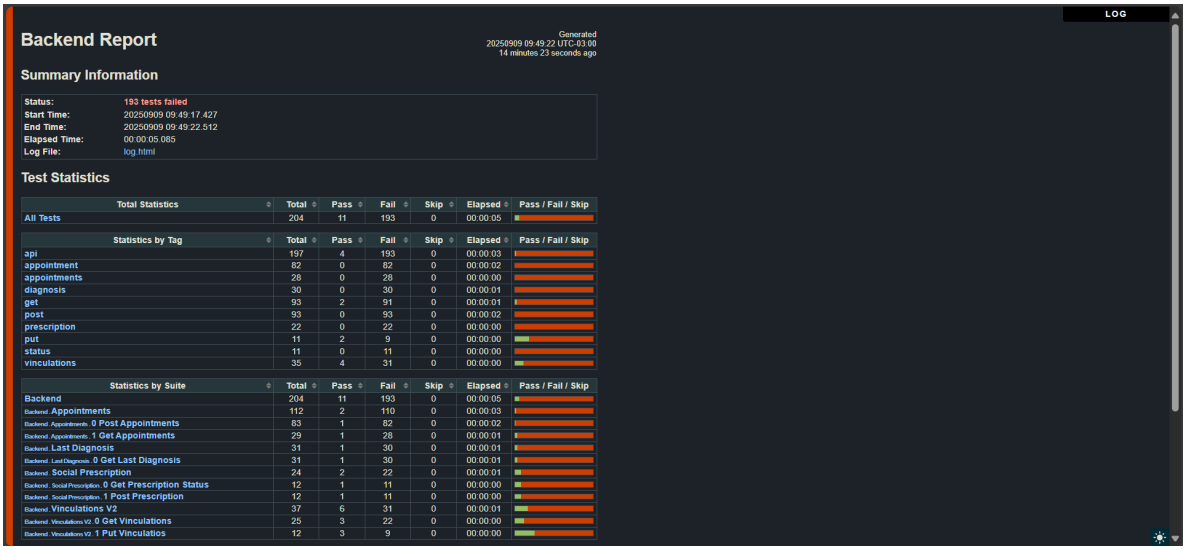


Figura B.8: Fragmento del reporte de pruebas generado por Robot Framework

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         variables.py

*** Variables ***
${CREDENTIALS_PATH}       ../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /vinculations

*** Test Cases ***
test-api-0 Get Token
    [Documentation]    Obtener el token para hacer consultas a la API
    Authenticate
    Log To Console    Token obtenido: ${TOKEN}

test-api-1 Get Vinculations with Status ACCEPTED for ID 18.475.065-1
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    vinculations    get

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}

    # Prepare query parameters based on test case input
    ${params}    Create Dictionary
    ...    patient_status=${TEST_CASE_1_INPUT["patient_status"]}

    # Make GET request to vinculations endpoint
    ${response}    GET On Session    api_session    url=${ENDPOINT}/${TEST_CASE_1_INPUT["id"]}    params=${params}    headers=${headers}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response contains expected output type
    Log To Console    Expected: ${TEST_CASE_1_EXPECTED_OUTPUT}
    Log To Console    Test case 1 completed successfully
```

Figura B.9: Fragmento del procedimiento de prueba de get vinculations.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         variables_put.py

*** Variables ***
${CREDENTIALS_PATH}       ../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /vinculation

*** Test Cases ***
test-api-0 Get Token···

test-api-1 Update vinculation status to REJECTED for patient ID 1
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    vinculations    put

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make PUT request to vinculation endpoint
    ${response}    PUT On Session    api_session    url=${ENDPOINT}/${TEST_CASE_1_INPUT["id"]}    json=${TEST_CASE_1_INPUT["request_body"]}    headers=${headers}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # For status 200, verify all fields except updated_at (which should be current date)
    ${response_json}    Set Variable    ${response.json()}
    Should Be Equal As Numbers    ${response_json["patient_id"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["patient_id"]}
    Should Be Equal As Numbers    ${response_json["doctor_id"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["doctor_id"]}
    Should Be Equal As Strings    ${response_json["status"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["status"]}
    Should Be Equal As Strings    ${response_json["created_at"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["created_at"]}

    # Verify response contains expected output type
    Log To Console    Expected: ${TEST_CASE_1_EXPECTED_OUTPUT}
    Log To Console    Test case 1 completed successfully
```

Figura B.10: Fragmento del procedimiento de prueba de put vinculations.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         variables_post.py

*** Variables ***
${CREDENTIALS_PATH}       ../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /appointment

*** Test Cases ***
test-api-0 Get Token
    [Documentation]    Obtener el token para hacer consultas a la API
    Authenticate
    Log To Console    Token obtenido: ${TOKEN}

test-api-1-1 Create appointment with valid data      You, 24 minutes ago • add post appointment suite test
    [Documentation]    ${TEST_CASE_1_1_DESCRIPTION}
    [Tags]    api    appointment    post

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make POST request to appointment endpoint
    ${response}    POST On Session    api_session    url=${ENDPOINT}    json=${TEST_CASE_1_1_INPUT["request_body"]}    headers=${headers}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_1_HTTP_STATUS}

    # For status 201, verify all fields except id (which should be an incremential id)
    ${response_json}    Set Variable    ${response.json()}

    # Verify response contains expected output fields
    Should Be Equal    ${response_json["healthcare_professional_id"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["healthcare_professional_id"]}
    Should Be Equal    ${response_json["patient_id"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["patient_id"]}
    Should Be Equal    ${response_json["reason"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["reason"]}
    Should Be Equal    ${response_json["notes"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["notes"]}
    Should Be Equal    ${response_json["dsm5"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["dsm5"]}
    Should Be Equal    ${response_json["cie11"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["cie11"]}
    Should Be Equal As Numbers    ${response_json["eeag"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["eeag"]}
    Should Be Equal    ${response_json["date"]}    ${TEST_CASE_1_1_EXPECTED_OUTPUT["date"]}

    Log To Console    Test case 1-1 completed successfully

test-api-1-2 Create appointment with invalid healthcare professional ID (invalid RUT)
```

Figura B.11: Fragmento del procedimiento de prueba de post appointments.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         variables_get.py

*** Variables ***
${CREDENTIALS_PATH}       ../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /appointments

*** Test Cases ***
test-api-0 Get Token···

test-api-1 Retrieve appointments with valid IDs
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    appointments    get

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make GET request to appointments endpoint
    ${response}    GET On Session    api_session    url=${ENDPOINT}/${TEST_CASE_1_INPUT["doctor_id"]}?patient_id=${TEST_CASE_1_INPUT["patient_id"]}    headers=${headers}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}
        # For status 200, verify the response structure
    ${response_json}    Set Variable    ${response.json()}

    # Verify response contains appointments list
    Dictionary Should Contain Key    ${response_json}    appointments
    ${appointments_list}    Set Variable    ${response_json["appointments"]}

    # Expected number of appointments: 2
    Length Should Be    ${appointments_list}    2

    # Validate each appointment in the response
    FOR    ${appointment}    IN    @{appointments_list}
        # Verify healthcare_professional_id matches doctor_id sent
        Should Be Equal    ${appointment["healthcare_professional_id"]}    ${TEST_CASE_1_INPUT["doctor_id"]}
        # Verify patient_id matches patient_id sent
        Should Be Equal    ${appointment["patient_id"]}    ${TEST_CASE_1_INPUT["patient_id"]}
    END

    Log To Console    Test case 1 completed successfully

test-api-2 Retrieve appointments with invalid healthcare professional ID (invalid RUT)
```

Figura B.12: Fragmento del procedimiento de prueba de get appointments.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         variables_get.py

*** Variables ***
${CREDENTIALS_PATH}       ../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /last_diagnosis

*** Test Cases ***
test-api-0 Get Token···

test-api-1 Get last diagnosis with valid healthcare professional ID and patient ID
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    diagnosis    get

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make GET request to last diagnosis endpoint
    ${response}    GET On Session    api_session    url=${ENDPOINT}?doctor_id=${TEST_CASE_1_INPUT["doctor_id"]}&patient_id=${TEST_CASE_1_INPUT["patient_id"]}    headers=${headers}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}
        # For status 200, verify the diagnosis response structure
    ${response_json}    Set Variable    ${response.json()}

    # Verify response contains diagnosis information
    Dictionary Should Contain Key    ${response_json}    dsm5
    Dictionary Should Contain Key    ${response_json}    cie11
    Dictionary Should Contain Key    ${response_json}    eeag

    # Validate diagnosis fields match expected output
    Should Be Equal    ${response_json["dsm5"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["dsm5"]}
    Should Be Equal    ${response_json["cie11"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["cie11"]}
    Should Be Equal As Numbers    ${response_json["eeag"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["eeag"]}

    Log To Console    Test case 1 completed successfully

test-api-2 Get last diagnosis with invalid healthcare professional ID (invalid RUT)
```

Figura B.13: Fragmento del procedimiento de prueba de get last diagnosis.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         ${CURDIR}/variables_get.py

*** Variables ***
${CREDENTIALS_PATH}       ${CURDIR}/../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /prescription/status

*** Test Cases ***
test-api-0 Get Token ···

test-api-1 Get prescription status for patient with valid ID
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    prescription    status    get

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make GET request to prescription status endpoint
    ${response}    GET On Session    api_session    url=${ENDPOINT}/${TEST_CASE_1_INPUT["user_id"]}    headers=${headers}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}

    # For status 200, verify the prescription status response structure
    ${response_json}    Set Variable    ${response.json()}

    # Verify response contains prescription status information
    Dictionary Should Contain Key    ${response_json}    user_id
    Dictionary Should Contain Key    ${response_json}    ready_for_prescription
    Dictionary Should Contain Key    ${response_json}    ready_to_answer
    Dictionary Should Contain Key    ${response_json}    count_answered_required_questions
    Dictionary Should Contain Key    ${response_json}    days_since_last_prescription
    Dictionary Should Contain Key    ${response_json}    has_prescription

    # Validate prescription status fields match expected output
    Should Be Equal As Numbers    ${response_json["user_id"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["user_id"]}
    Should Be Equal    ${response_json["ready_for_prescription"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["ready_for_prescription"]}
    Should Be Equal    ${response_json["ready_to_answer"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["ready_to_answer"]}
    Should Be Equal As Numbers    ${response_json["count_answered_required_questions"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["count_answered_required_questions"]}
    Should Be Equal As Numbers    ${response_json["days_since_last_prescription"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["days_since_last_prescription"]}
    Should Be Equal    ${response_json["has_prescription"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["has_prescription"]}

    Log To Console    Test case 1 completed successfully
```

Figura B.14: Fragmento del procedimiento de prueba de get prescription status.robot

```
*** Settings ***
Library           RequestsLibrary
Library           Collections
Library           JSONLibrary
Variables         ${CURDIR}/variables_post.py

*** Variables ***
${CREDENTIALS_PATH}       ${CURDIR}/../credentials.json
${TOKEN}                  None
${API_URL}                http://backend-test:8000
${ENDPOINT}               /prescription

*** Test Cases ***
test-api-0 Get Token ···

test-api-1 Post prescription for a valid patient ID
    [Documentation]    ${TEST_CASE_1_DESCRIPTION}
    [Tags]    api    prescription    post

    # Create session for API calls
    Create Session    api_session    ${API_URL}

    # Set up headers with authentication token
    ${headers}    Create Dictionary    Authorization=Bearer ${TOKEN}    Content-Type=application/json

    # Make POST request to prescription endpoint with user_id as path parameter
    ${response}    POST On Session    api_session    url=${ENDPOINT}/${TEST_CASE_1_INPUT["user_id"]}    headers=${headers}

    # Log status codes to console
    Log To Console    Expected Status Code: ${TEST_CASE_1_HTTP_STATUS}
    Log To Console    Actual Status Code: ${response.status_code}

    # Verify response status code
    Should Be Equal As Numbers    ${response.status_code}    ${TEST_CASE_1_HTTP_STATUS}

    # For status 201, verify the prescription response structure
    ${response_json}    Set Variable    ${response.json()}

    # Verify response contains prescription information
    Dictionary Should Contain Key    ${response_json}    user_id
    Dictionary Should Contain Key    ${response_json}    activity_id
    Dictionary Should Contain Key    ${response_json}    active
    Dictionary Should Contain Key    ${response_json}    assists
    Dictionary Should Contain Key    ${response_json}    not_assists
    Dictionary Should Contain Key    ${response_json}    created_at
    Dictionary Should Contain Key    ${response_json}    updated_at
    Dictionary Should Contain Key    ${response_json}    is_prescription
    Dictionary Should Contain Key    ${response_json}    id
    Dictionary Should Contain Key    ${response_json}    activity

    # Validate prescription fields match expected output
    Should Be Equal As Numbers    ${response_json["user_id"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["user_id"]}
    Should Be Equal As Numbers    ${response_json["activity_id"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["activity_id"]}
    Should Be Equal    ${response_json["active"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["active"]}
    Should Be Equal As Numbers    ${response_json["assists"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["assists"]}
    Should Be Equal As Numbers    ${response_json["not_assists"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["not_assists"]}
    Should Be Equal    ${response_json["is_prescription"]}    ${TEST_CASE_1_EXPECTED_OUTPUT["is_prescription"]}

    # Validate activity information
    Dictionary Should Contain Key    ${response_json["activity"]}    name
    Dictionary Should Contain Key    ${response_json["activity"]}    description
    Dictionary Should Contain Key    ${response_json["activity"]}    address
    Dictionary Should Contain Key    ${response_json["activity"]}    city
    Dictionary Should Contain Key    ${response_json["activity"]}    activity_type

    Log To Console    Test case 1 completed successfully
```

Figura B.15: Fragmento del procedimiento de prueba de post prescription.robot

Bibliografía

[1] American Psychiatric Association. *DSM-5 Update*. https://psychiatryonline.org/pb-assets/dsm/update/DSM5Update_octubre2018_es.pdf. Disponible en línea, octubre de 2018. 2018.

[2] Atlassian. *Software Testing*. Accessed: 2025-08-05. Atlassian. 2025. URL: https://www.atlassian.com/continuous-delivery/software-testing.

[3] *BS ISO 81001-1:2021 – Health Software and Health IT Systems Safety, Effectiveness and Security: Principles and Concepts*. ISO/TC 215/JWG 7. Identical to ISO 81001-1:2021. Publication date: 10 May 2021. British Standards Institution (BSI), mayo de 2021.

[4] Jean Endicott et al. "The Global Assessment Scale: A procedure for measuring overall severity of psychiatric disturbance". En: *Archives of General Psychiatry* 33.6 (1976), págs. 766-771. DOI: 10.1001/archpsyc.1976.01770060086012.

[5] Vahid Garousi et al. "Software-testing education: A systematic literature mapping". En: *Journal of Systems and Software* 165 (2020), pág. 110570. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110570. URL: https://www.sciencedirect.com/science/article/pii/S0164121220300510.

[6] GitHub. *What is Software Testing? Understanding Software Testing*. Accessed: 2025-08-05. GitHub. Jul. de 2024. URL: https://github.com/resources/articles/software-development/what-is-software-testing.

[7] IEEE Computer Society. *Software Architecture*. Recuperado el 11 de agosto de 2025. 2025. URL: https://www.computer.org/resources/software-architecture.

[8] International Software Testing Qualifications Board. *Black-Box Testing — ISTQB Glossary*. Accessed: 2025-08-07. 2025. URL: https://glossary.istqb.org/en_US/term/black-box-testing.

[9] International Software Testing Qualifications Board. *Compliance — ISTQB Glossary*. Accessed: 2025-08-07. 2025. URL: https://glossary.istqb.org/en_US/term/compliance.

[10] International Software Testing Qualifications Board. *Defect — ISTQB Glossary*. Accessed: 2025-08-05. 2024. URL: https://glossary.istqb.org/en_US/term/defect.

[11] International Software Testing Qualifications Board. *Error — ISTQB Glossary*. Accessed: 2025-08-05. 2024. URL: https://glossary.istqb.org/en_US/term/error.

[12] International Software Testing Qualifications Board. *Experience-Based Testing — ISTQB Glossary*. Accessed: 2025-08-07. 2025. URL: https://glossary.istqb.org/en_US/term/experience-based-testing.

[13] International Software Testing Qualifications Board. *Failure — ISTQB Glossary*. Accessed: 2025-08-05. 2024. URL: https://glossary.istqb.org/en_US/term/failure.

[14] International Software Testing Qualifications Board. *Test Item — ISTQB Glossary*. Accessed: 2025-08-07. 2025. URL: https://glossary.istqb.org/en_US/term/test-item.

[15] International Software Testing Qualifications Board. *Testing — ISTQB Glossary*. Accessed: 2025-08-14. 2025. URL: https://glossary.istqb.org/en_US/term/testing.

[16] International Software Testing Qualifications Board. *White-Box Testing — ISTQB Glossary*. Accessed: 2025-08-07. 2025. URL: https://glossary.istqb.org/en_US/term/white-box-testing.

[17] ISO/IEC/IEEE. *ISO/IEC/IEEE 24765:2017 Systems and Software Engineering — Vocabulary*. Standard. International Organization for Standardization, International Electrotechnical Commission, y IEEE, 2017.

[18] *ISO/IEC/IEEE 29119-2:2021 Software and systems engineering — Software testing — Part 2: Test processes*. Standard No. ISO/IEC/IEEE 29119-2:2021. Geneva, Switzerland: International Organization for Standardization; International Electrotechnical Commission; Institute of Electrical y Electronics Engineers, 2021.

[19] *ISO/IEC/IEEE 29119-4:2021 Software and systems engineering — Software testing — Part 4: Test techniques*. Standard No. ISO/IEC/IEEE 29119-4:2021. Geneva, Switzerland: International Organization for Standardization; International Electrotechnical Commission; Institute of Electrical y Electronics Engineers, 2021.

[20] Gerard Meszaros. *xUnit Test Patterns: Refactoring Test Code*. Includes bibliographical references and index. Upper Saddle River, NJ: Pearson Education, 2007. ISBN: 978-0-13-149505-0.

[21] C. Muhl et al. "Establishing internationally accepted conceptual and operational definitions of social prescribing through expert consensus: a Delphi study". En: *BMJ Open* 13 (2023), e070184. DOI: 10.1136/bmjopen-2022-070184.

[22] National Academy for Social Prescribing. *What is social prescribing?* Consultado: 5 de agosto de 2025. 2025. URL: https://socialprescribingacademy.org.uk/what-is-social-prescribing/.

[23] NHS England. *Social prescribing*. Consultado: 5 de agosto de 2025. 2025. URL: https://www.england.nhs.uk/personalisedcare/social-prescribing/.

[24] Object Management Group. *OMG Unified Modeling Language (OMG UML), Version 2.5.1*. Inf. téc. formal/2017-12-05. Normative machine-readable files: https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi, https://www.omg.org/spec/UML/20161101/UML.xmi, https://www.omg.org/spec/UML/20161101/StandardProfile.xmi, https://www.omg.org/spec/UML/20161101/UMLDI.xmi. Object Management Group (OMG), dic. de 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF.

[25] Open University. *Social prescribing*. Consultado: 5 de agosto de 2025. 2025. URL: https://www.open.edu/openlearn/mod/oucontent/view.php?id=166863&section=_unit5.3.2.2.

[26] Organización Mundial de la Salud. *Guía de Referencia de la CIE-11*. Sección 1.2.4 Características generales de la CIE-11, p. 35. Organización Mundial de la Salud. 2025. URL: https://icdcdn.who.int/static/releasefiles/2025-01/ICD-11-Reference-Guide-2025-01-es.pdf.

[27] Saheed Popoola, Vineela Kunapareddi y Hazem Said. "Developing and sustaining a student-driven software solutions center—An experience report". En: *Journal of Systems and Software* 220 (2025), pág. 112279. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2024.112279. URL: https://www.sciencedirect.com/science/article/pii/S0164121224003236.

[28] Jill Sonke et al. "Social prescribing outcomes: a mapping review of the evidence from 13 countries to identify key common outcomes". En: *Frontiers in Medicine* 10 (2023), pág. 1266429. DOI: 10.3389/fmed.2023.1266429.

[29] Stanford Lifestyle Medicine. *What is social prescribing and why is it important?* Consultado: 5 de agosto de 2025. 2025. URL: https://lifestylemedicine.stanford.edu/what-is-social-prescribing-and-why-is-it-important/.

[30] Stephanie Susnjara y Ian Smalley. *What is Software Testing?* Accessed: 2025-08-05. IBM. Jul. de 2025. URL: https://www.ibm.com/think/topics/software-testing.

[31] Universidad Técnica Federico Santa María. *Programa de Asignatura: Gestión de Proyectos de Informática (INF-360)*. Departamento de Informática, UTFSM. Aprobado por Acuerdo CC.DD. 13/2016. 2016.

[32] Universidad Técnica Federico Santa María. *Programa de Asignatura: Taller de Desarrollo de Informática (INF-288)*. Departamento de Informática, UTFSM. Aprobado por Acuerdo CC.DD. 13/2016. 2016.

[33] vFunction. *What is Software Architecture?* Recuperado el 11 de agosto de 2025. 2023. URL: https://vfunction.com/blog/what-is-software-architecture/.

[34] Visual Paradigm. *What is Sequence Diagram?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/. Recuperado el 9 de septiembre de 2025. s.f.

[35] Visual Paradigm. *What is Use Case Diagram?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/. Recuperado el 9 de septiembre de 2025. s.f.

[36] Hironori Washizaki, ed. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)*. Version 4.0. IEEE Computer Society, 2024. URL: https://www.swebok.org.

[37] World Health Organization Regional Office for the Western Pacific. *A Toolkit on How to Implement Social Prescribing*. Licence: CC BY-NC-SA 3.0 IGO. Manila: World Health Organization Regional Office for the Western Pacific, 2022. URL: https://iris.who.int/bitstream/handle/10665/354456/9789290619765-eng.pdf.