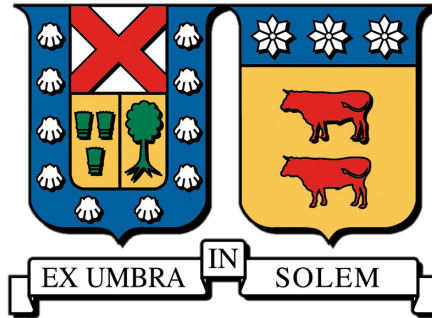


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



**“DESARROLLO DE PLATAFORMA WEB DE
MINUTAS DE ALIMENTACIÓN ESCALABLE EN
JARDÍN Y SALA CUNA NATGIO”**

JAVIER ALEJANDRO SANHUEZA MEJÍAS

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL TELEMÁTICO**

PROFESOR GUÍA: MAURICIO ARAYA

ENERO, 2023

Agradecimientos

En primer lugar quiero agradecer al creador de todas las cosas, Dios. El cual me ha dado la oportunidad de nacer, me ha acompañado en todos los momentos de mi vida y me ha ayudado en cada dificultad que he tenido por guardar el día de reposo, el día Sábado. Agradecer a mi familia, papá, mamá y hermano, gracias por darme los medios económicos para poder ser profesional, y por darme aliento cada vez que las cosas se ponían difíciles. Agradecer a cada uno de las personas que han pasado por mi vida universitaria, profesores, compañeros y amigos. Finalmente, agradecer a mi nueva familia, Iveth y Theito, gracias por su amor incondicional, por sus risas y cariños que me motivan a ser un excelente profesional.

Javier Alejandro Sanhuesa Mejías

Resumen

El Jardín Infantil y Sala Cuna Natgio, ha decidido actualizar su sistema de creación de minutas, mediante una plataforma web, ya que en la actualidad la información y creación de las minutas se realiza y almacena en un archivo.xlsx.

El siguiente trabajo de título, tiene como finalidad describir la problemática que hoy enfrenta la administración del jardín para la generación de minutas. Se analiza su forma actual de trabajo, se presenta una solución y se implementa la solución en la web.

Como solución, se propone la creación de una aplicación web con un sistema de usuarios, en la que se puedan crear minutas a partir de recetas según los niveles pre-escolares del jardín.

La idea principal de la aplicación es que el usuario pueda ingresar los ingredientes necesarios para poder crear una receta en un determinado nivel pre-escolar. Con estas recetas es posible crear una minuta para un día específico.

La aplicación constará de algunos requisitos funcionales, como por ejemplo: 3 tipos de usuarios (administrador, nutricionista y manipulador), generación de informes PDF, creación de ingredientes, creación de sucursales, informe de stock de ingredientes, entre otros.

Se detalla la arquitectura de la plataforma, la cual consiste en un frontend, desarrollado en Quasar (basado en Vue.js); un backend, dividido en una API desarrollada en Express.js y una base de datos relacional con un motor en MySQL.

Además de analizar las vistas creadas de la aplicación, se muestra la implementación en producción en los servidores de la agencia Ingenium y Heroku.

keywords: Aplicación web, jardín infantil, minutas alimenticias, recetas, ingredientes, stock.

Abstract

The Natgio Kindergarten and Nursery Room, has decided to update its system for the creation of menus through a web platform, since currently the information and creation of the menus are made and stored in an xlsx file.

This finale Project has the purpose of describing the problems that today faces the administration of this institution for the production of menus. Their current way of working is analyzed, a solution is presented, and the solution is implemented on the web.

As a solution, the creation of a web application with a user system is proposed, in which the menus can be created from recipes according to the pre-school levels of the institution.

The main idea of the app is that the user can put the necessary ingredients and doing so, this app would be able to create a recipe at a certain pre-school level. With these recipes it is possible to create a menu for a specific day.

This app will have some functional requirements, such as: 3 types of users (administrator, nutritionist and handler), PDF report generation. creation of ingredients, creation of Branch offices, report of the stock of ingredients, among others.

The architecture of the platform is detailed, which consists of a frontend, development made on Quasar (based on Vue.js); a backend, divided into an API developed in Express.js and a relational database with a MySQL engine.

Besides that, it's able to analyzing the created views of the application, the implementation is shown. in production on the servers of the Ingenium and Heroku agency.

keywords: Web application, kindergarten, nutritional minutes, recipes, ingredients, stock.

Glosario

Agencia Ingenium: agencia de marketing digital, publicidad, diseño e implementación de página web a través de Wordpress.

API: conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Aplicación web: conjunto de páginas dinámicas cuyo contenido se determina después que un usuario haya interactuado con ella.

Backend: parte del desarrollo web que se encarga de que toda la lógica de una aplicación web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

cPanel: plataforma que permite administrar sitios web que se encuentran dentro de un hosting.

Framework: marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software.

Frontend: parte del sitio web con la que el usuario interactúa. Incluye todo el contenido que se muestra y que es visible para el público o los usuarios que han iniciado sesión.

HTML: lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, entre otros.

HTTP: lenguaje que media entre las peticiones del cliente y las respuestas del servidor en la web, para permitir una comunicación fluida y en un mismo “lenguaje”.

JSON: formato de datos que se basa en un texto que respeta la sintaxis de objeto de JavaScript.

JWT: token de acceso estandarizado en el RFC 7519 que permite el intercambio seguro de datos entre dos partes.

Local Storage: objeto que permite almacenar datos de manera local en el navegador y sin necesidad de realizar alguna conexión a una base de datos.

Material guidelines: guía completa para el diseño visual, interactivo y de movimiento en plataformas y dispositivos, concebido para unificar el aspecto de la interfaz de Google.

Middleware: software con el que las diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar las aplicaciones de manera inteligente y eficiente, de forma que se pueda innovar más rápido.

Minutas de alimentación: conjunto de alimentos o preparaciones que están consideradas en un tiempo de comida. Es decir, qué se comerá para el desayuno, almuerzo o cena.

Natgio: es un establecimiento de educación pre-escolar particular ubicado en la ciudad de Quilpué, Valparaíso. Abre sus puertas en el año 2014 y desde entonces ha contribuido con el desarrollo integral de cientos de niños y niñas, posicionándonos así como uno de los jardines infantiles más prestigiosos de la zona.

PDF: formato de archivo universal que conserva las fuentes, las imágenes y la maquetación de los documentos originales creados en una amplia gama de aplicaciones y plataformas.

Stakeholders: son aquellas personas y entidades que tienen algún impacto o interés en un proyecto.

Receta: texto instructivo que entrega las indicaciones e ingredientes para preparar un plato.

Requisitos funcionales: son aquellos que describen cualquier actividad que debe realizar un sistema, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

Requisitos no funcionales: explican los aspectos de calidad del sistema que se va a construir, por ejemplo: rendimiento, portabilidad, usabilidad, entre otros.

Token: string aleatorio que identifica a un usuario y puede ser utilizado por la aplicación para realizar llamadas API.

URL: dirección única y específica que se asigna a cada uno de los recursos disponibles de la World Wide Web para que puedan ser localizados por el navegador y visitados

por los usuarios.

Acrónimos

API: Application Programming Interface

HTML: HyperText Markup Language

HTTP: Hypertext Transfer Protocol

JSON: JavaScript Object Notation

JWT: JSON Web Token

MVC: Model View Controller

MVP: Minimum Viable Product

PDF: Portable Document Format

QR: Quick Response

SQL: Structured Query Language

URL: Uniform Resource Locator

WHM: Web Host Manager

Índice de figuras

2.1. Hoja de cálculo de base de datos para minutas	16
2.2. Hoja de cálculo de minutas Julio 2021	17
2.3. Hoja de cálculo parámetros	17
2.4. Hoja de cálculo de parámetros de minutas	18
4.1. Diagrama de arquitectura de la plataforma	40
4.2. Diagrama Entidad - Relación de la base de datos, donde la entidad usuarios interactúa con las demás entidades que participan en la crea- ción de minutas.	42
4.3. Diagrama de flujo de la vista login.	47
4.4. Diagrama de flujo guardar minutas en PDF	49
4.5. Diagrama de flujo guardar calendario en PDF	49
4.6. Diagrama de flujo del componente tabla variables	52
4.7. Diagrama de flujo formulario creación variables	53
4.8. Diagrama de flujo de las acciones de tabla variables	54
4.9. Diagrama de flujo de la vista copiar minutas	56
5.1. Subdominios de natgio.cl	60
5.2. Base de datos de la plataforma a través de phpMyAdmin	61
5.3. Backend de la plataforma alojado en Heroku	61
5.4. Reporte de la disponibilidad y tiempo de respuesta de la página donde está alojada la plataforma.	62

5.5. Reporte de la disponibilidad y tiempo de respuesta de la API en Heroku.	63
5.6. Vista Home de la aplicación.	64
5.7. Modal creación de minutas.	65
5.8. Modal visualización de minutas.	65
5.9. Minutas en PDF.	66
5.10. Vista listado de compra.	66
5.11. Calendario de minutas por nivel en PDF.	67
A.1. Diagrama de flujo de los Token	74
D.1. Diagrama de flujo de la vista selección sucursal	82
D.2. Diagrama de flujo del componente de calendario	83
D.3. Diagrama de flujo para crear minutas	86
D.4. Diagrama de flujo detalles minutas	87
D.5. Diagrama de flujo cambiar nivel	88
D.6. Diagrama de flujo lista de compra	89
D.7. Diagrama de flujo actualizar matrículas	91
D.8. Diagrama de flujo registro de un usuario.	93
D.9. Diagrama de flujo del componente tabla de usuarios	96
D.10. Diagrama de flujo del acciones del componente tabla de usuarios	97
D.11. Diagrama de flujo de las acciones de tabla stock	99
D.12. Diagrama de flujo de la acción filtra por nivel escolar en tabla matrículas	101
D.13. Diagrama de flujo de la vista editar mis datos	103
D.14. Diagrama de flujo de la vista editar mis datos	105
D.15. Diagrama de flujo de la acción cerrar sesión	106
D.16. Vista del Login.	107
D.17. Vista selección de sucursal.	108
D.18. Vista Registro de usuario en sucursal.	108
D.19. Vista Registro de usuario en sucursal.	109
D.20. Vista Creación variables institucionales	110

D.21. Vista Creación variables alimentarias	111
D.22. Modal confirmación de eliminación de dato	111
D.23. Vista Stock	112
D.24. Vista Matrículas	113
D.25. Vista Copiar minutas	113
D.26. Vista Recetas	114
D.27. Vista Editar Receta	115
D.28. Vista Editar mis datos	116

Índice general

1. Introducción	7
1.1. Objetivos	8
1.1.1. Objetivo General	8
1.1.2. Objetivos Específicos	8
1.2. Trabajo realizado	8
1.3. Estructura de la memoria	10
2. Problemática	11
2.1. Minutas Alimenticias en Jardín Infantil Natgio	11
2.1.1. Niveles pre-escolares	11
2.1.2. Tipos de ingredientes	12
2.1.3. Recetas	12
2.1.4. Tipos de colaciones	13
2.1.5. Jornadas	13
2.1.6. Matrículas	13
2.2. Proceso de Creación de Minutas en Jardín Infantil Natgio	14
3. Marco tecnológico	19
3.1. DietPlan	19
3.2. Food Processor	20
3.3. Menutech	20
3.4. Nutrigreen	21

3.5. Minuta Virtual	22
3.6. Tecnologías	23
3.6.1. Lenguaje de programación	23
3.6.2. Tecnologías para Frontend	24
3.6.3. Tecnologías para Backend	28
3.6.4. Base de datos	31
4. Desarrollo de la plataforma	35
4.1. Requerimientos de la plataforma	35
4.1.1. Requisitos funcionales	36
4.1.2. Requerimientos no funcionales	38
4.2. Tecnologías a ocupar	38
4.2.1. Base de datos	39
4.2.2. Frameworks	40
4.3. Base de datos de la aplicación	41
4.4. Backend de la aplicación	43
4.5. Frontend de la aplicación	45
4.5.1. Login de usuarios	45
4.5.2. Vista Home	48
4.5.3. Menú de Navegación	50
4.6. Resumen de vistas	56
5. Resultados	59
5.1. Implementación	59
5.2. Disponibilidad de la plataforma	62
5.3. Imágenes de la aplicación web	63
5.3.1. Vista Home	63
6. Conclusiones y Trabajos futuros	69

Anexos	71
A. Definición y funcionamiento de los tokens	73
B. Definición y funcionamiento de Vuex	75
C. Endpoints de la aplicación	77
D. Detalles y diagramas de flujo	81
D.1. Vista Selección de sucursal	81
D.2. Vista Home	82
D.2.1. Cambiar de nivel	88
D.2.2. Lista de compra	88
D.2.3. Cantidad de matrículas	89
D.3. Menú de navegación	92
D.3.1. Registrar usuario	92
D.3.2. Listar usuarios	94
D.3.3. Stock	97
D.3.4. Matrículas	99
D.3.5. Receta	102
D.3.6. Editar mis datos	104
D.3.7. Cerrar Sesión	106
D.4. Imágenes de la aplicación	107
D.5. Vista Login	107
D.6. Selección de sucursal	107
D.7. Vista Registrar	108
D.8. Vista Listar usuarios	109
D.9. Vista Creación variables institucionales	109
D.10. Vista Creación variables alimentarias	110
D.11. Vista Stock	111

D.12. Vista Matrículas	112
D.13. Vista Copiar minutas	113
D.14. Vista Recetas	114
D.15. Vista Editar mis datos	115

Capítulo 1

Introducción

El desarrollo de Software se ha convertido; a lo largo de los años, en una solución ajustada a las necesidades de millones de empresas, del país y del mundo. Una rama del desarrollo de software es el desarrollo web, el cual crea software orientado al usuario en el contexto de los sitios web. La aprobación de esta tecnología, ha captado el interés de los usuarios, convirtiéndolos en posibles futuros mandantes. A través del desarrollo web cada mandante o cliente puede acceder a sus plataformas en cualquier parte del mundo y el cualquier dispositivo con conexión a internet.

El Jardín Infantil y Sala Cuna Natgio. ha aprovechado los beneficios que ofrece el uso de esta herramienta, solicitando un sistema de minuta alimenticia, en el que deberá acceder el administrador, manipulador de alimentos y nutricionista del establecimiento. Es por esta razón, que se inicia una recopilación de información, evaluando 8 tecnologías para la selección de la más adecuada.

En la presente memoria, se expone el trabajo desarrollado para el sistema de minuta del Jardín Infantil y Sala Cuna Natgio.

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar una plataforma web para la creación, visualización y actualización de minutas de alimentación para jardín infantil.

1.1.2. Objetivos Específicos

- Conceptualizar el proceso de minutas alimenticias en Jardín Infantil y Sala Cuna Natgio.
- Optimizar el tiempo de creación de las minutas.
- Crear interfaz cómoda e intuitiva para el usuario.
- Implementar los requerimientos planteados por la contra-parte para el proceso de creación de minutas.

1.2. Trabajo realizado

Para ejecutar este proyecto es necesario seguir en orden las siguientes etapas:

■ Definir los requerimientos de la aplicación

Los requerimientos del software son las necesidades que tienen las partes interesadas del software (stakeholders) para que el sistema cumpla de manera satisfactoria su función. Esto se logra mediante reuniones frecuentes hasta definir el último requerimiento. Hay que tener en cuenta que durante el desarrollo del software los requerimientos pueden ir cambiando.

■ Definir perfiles de usuarios de la plataforma

Dentro de la aplicación web hay 3 perfiles, los cuales son:

- Administrador

- Nutricionista
- Manipuladora de alimentos.

Por lo tanto, fue necesario definir qué funciones y permisos tiene cada uno de los perfiles antes mencionados.

- **Investigar y definir tecnologías en concordancia a funcionalidades requeridas y compatibilidad entre ellas.**

Hoy en día, existen variadas opciones al momento de desarrollar aplicaciones web, ya sea en el lenguaje de programación a utilizar, los framework, tipo de base de datos, entre otros. Por lo cual, se realizó un estudio para definir qué elementos de los comentados anteriormente se elegirán, dependiendo de los requerimientos que se definirán.

- **Desarrollar backend y base de datos de la plataforma para la conexión de las distintas funcionalidades.**

Para desarrollar el backend se utilizó el framework Express, el cual está basado en Nodejs, el que permite trabajar con APIs para realizar consultas con verbos HTTP en la base de datos. Para la base de datos se utiliza una de tipo relacional, la cual permite representar datos en tablas de forma intuitiva y directa. Para la gestión de la base de datos se ocupa MySQL.

- **Desarrollar frontend para la plataforma para el envío y recepción de información desde backend.**

Para desarrollar el frontend de la aplicación web se utilizó el framework Quasar, el cual está basado en Vuejs. Quasar permite de manera rápida realizar una interfaz web de última generación basada en Material Guidelines, la cual es utilizada por Google [1]. Material Guidelines describe cómo asegurarse de que la aplicación sea utilizable para todos los usuarios.

- **Implementar el Software.**

Finalmente, se desplegó la aplicación en distintas plataformas para que la apli-

cación web pueda funcionar de manera continua. Para el backend se utiliza la plataforma Heroku, la cual es una plataforma de servicios en la nube que permite manejar los servidores y sus configuraciones, escalamiento y la administración. El frontend se aloja en los servidores de la Agencia Ingenium (agencia encargada del proyecto). Los servidores ocupan cPanel (panel de control para administrar servidores de alojamiento web que proveen herramientas de automatización y una interfaz gráfica basada en páginas web).

1.3. Estructura de la memoria

Este documento se estructura de la siguiente forma: en el Capítulo 2, se hace una revisión del proceso de creación de minutas que realizaba el jardín antes del desarrollo de la plataforma; en el Capítulo 3, se presentan diversas aplicaciones y plataformas web de manejo de minutas alimenticias que realizan funciones similares a la de este proyecto; en el Capítulo 4 se desarrolla el plan de trabajo presentado en la Sección 1.2 presenta diversas herramientas para el desarrollo de aplicaciones web, comparaciones, ventajas y desventajas, además de los diagramas, arquitectura y componentes que conforman la aplicación desarrollada; mientras que en el capítulo 5, se describe el desarrollo final de la aplicación. Por último, en el capítulo 6, se mencionan las conclusiones y se detallan las futuras mejoras que se pueden llegar a implementar a la solución.

Capítulo 2

Problemática

Para las instituciones públicas o privadas que tengan un casino en sus dependencias, es crucial tener un buen sistema de minutas. Es importante mantener informados a los usuarios del casino sobre los menús disponibles. Además, estos sistemas ayudan a las instituciones a mantener de forma ordenada y actualizada el stock de los ingredientes que se utilizan para preparar los menús.

2.1. Minutas Alimenticias en Jardín Infantil Natgio

Como se mencionó en la introducción, esta plataforma es desarrollada para el Jardín Infantil y Sala Cuna Natgio. Para contextualizar la problemática, es necesario tener presente los conceptos usados por Natgio para poder entender el proceso de creación de minutas alimenticias.

2.1.1. Niveles pre-escolares

Dentro de Natgio se encuentran 3 niveles pre-escolares, los cuales agrupan a niños y niñas según su edad. Estos niveles se presentan en el Cuadro 2.1 con su respectivo rango de edad.

Desde el año 2014 estos niveles se encuentran funcionando en Natgio, pero es po-

Nivel escolar	Rango de edad
Sala Cuna Menor	entre 5 meses a 1 año
Sala Cuna Mayor	entre 1 a 2 años
Jardín Infantil	entre 2 a 5 años

Cuadro 2.1: Niveles pre-escolares en Natgio

sible que en un futuro se puedan ampliar a otros niveles pre-escolares.

2.1.2. Tipos de ingredientes

Para poder obtener el stock disponible de ingredientes para crear las minutas es necesario definir los dos tipos de ingredientes que se utilizan en el Jardín.

1. Ingrediente Supermercado: Son todos los ingredientes o productos que se compran en algún establecimiento comercial, ya sea supermercado, mayorista o minimarket. Como por ejemplo: huevos, aceite, yogur, leche, fideos, etc.
2. Ingrediente Perecible: Son todos los ingredientes o productos que son adquiridos en ferias o mercados. Como por ejemplo frutas y verduras.

Se hace esta distinción entre ingredientes debido a que los productos perecibles se adquieren cada una semana (todos los días miércoles). Mientras que los productos que se adquieren en un supermercado son comprados una vez al mes. Por lo cual, es necesario tener un stock semanal y mensual.

2.1.3. Recetas

Una minuta alimenticia está constituida por diferentes platos de comida, los cuales se preparan a partir de una receta. Esta receta se conforma de ingredientes ya sean de tipo supermercado o perecible. Según sea el nivel pre-escolar, la receta se verá modificada en la cantidad de cada ingredientes a usar. Las unidades de medidas que se ocupan son gramos [g], centímetros cúbicos [cc], cucharaditas y gotas.

2.1.4. Tipos de colaciones

Una minuta alimenticia diaria se divide en 3 tipos de colaciones: Colación AM, Almuerzo y Colación PM. A su vez, el almuerzo se divide en 4: Ensalada, Principal, Acompañamiento y Postre. El cuadro 2.2 muestra los horarios establecidos para ofrecer las colaciones a los niños y niñas según su nivel pre-escolar.

Nivel pre-escolar	Horario		
	Colación AM	Almuerzo	Colación PM
Sala Cuna Menor y Mayor	09:10 hrs	11:30 hrs	15:15 hrs
Jardín Infantil	09:00 hrs	12:10 hrs	16:15 hrs

Cuadro 2.2: Tipo de colaciones y horarios en Natgio

2.1.5. Jornadas

Los niños y niñas, dependiendo del nivel pre-escolar al que pertenecen, tienen distintos tipos de jornadas según la necesidad que tenga el apoderado del niño. Los cuadros 2.3 y 2.4 muestran las distintas jornadas y tipos de colaciones que reciben cada uno de los niños y niñas según su nivel pre-escolar.

Jornadas	Horario	Colaciones
Jornada 1	07:30 a 13:30	Colación AM y Almuerzo
Jornada 2	07:30 a 16:00	Colación AM y Almuerzo
Jornada 3	07:30 a 18:00	Colación AM, Almuerzo y Colación PM
Jornada 4	07:30 a 18:30	Colación AM, Almuerzo y Colación PM
Jornada 5	14:15 a 16:15	Sin colación
Jornada Extendida	07:30 a 19:00	Colación AM, Almuerzo y Colación PM

Cuadro 2.3: Jornadas para nivel Jardín Infantil en Natgio

2.1.6. Matrículas

Las matrículas o mensualidades son la cantidad de niños y niñas que hay en cada una de las jornadas. Con este número es posible calcular la cantidad total de ingredien-

Jornadas	Horario	Colaciones
Jornada 1	07:30 a 16:00	Colación AM, Almuerzo y Colación PM
Jornada 2	07:30 a 18:00	Colación AM, Almuerzo y Colación PM
Jornada 3	07:30 a 18:30	Colación AM, Almuerzo y Colación PM
Jornada Extendida	07:30 a 19:00	Colación AM, Almuerzo y Colación PM

Cuadro 2.4: Jornadas para nivel Sala Cuna Menor y Mayor en Natgio

tes que se necesitarán por día en cada una de las recetas, lo cual, permite calcular el stock semanal y mensual, como también la lista de compra de los ingredientes faltantes.

2.2. Proceso de Creación de Minutas en Jardín Infantil Natgio

Existen 3 usuarios que interactúan en el proceso de creación de minutos en Natgio. Estos son:

1. Administrador del Jardín: Su principal función, aparte de llevar todo lo que implica la administración de un jardín, es realizar las compras de los ingredientes (supermercado y perecibles), además de actualizar el stock de éstos.
2. Nutricionista: Es el encargado de crear las minutos y recetas con sus respectivas cantidades de ingredientes, dependiendo del nivel pre-escolar de los niños o niñas.
3. Manipulador de alimentos: Su función es preparar las minutos del día. Recibe un documento con el total de niños por cada tipo de colación que habrá en el día, para que de esta forma se ocupe la cantidad exacta de cada ingrediente.

El proceso de creación de minutos utilizado por Natgio, se hace a través de un archivo .xlsx (el cual se puede abrir con Excel o en Numbers para mac), el que debe

ser modificado mes a mes. Por otro lado, este archivo funciona como base de datos, en donde se encuentran 117 platos con sus respectivas recetas y 59 ingredientes. Cabe señalar, que el número de tipo de plato e ingredientes puede ir variando cada mes.

Para poder ingresar un nuevo plato, la nutricionista debe de ir a la última fila del archivo para ingresar la cantidad de cada producto por cada nivel. Además, define el tipo de colación y el tipo del ingrediente a ocupar. En la figura 2.1 se puede visualizar un ejemplo de cómo se ve la base de datos en el archivo que utiliza Natgio para crear las minutas.

Para poder crear las minutas de un mes en específico, la nutricionista en el mismo archivo encuentra el mes en el cual hará las minutas. Aquí ella define qué comerán los niños de cada nivel por cada día del mes. Además, la administradora del jardín define la cantidad de matrículas que existen por cada tipo de colación por cada nivel pre-escolar.

La figura 2.2 muestra un ejemplo de cómo se crean las minutas para el mes de Julio del año 2021 para el nivel Jardín Infantil. En esta misma hoja se encuentran las minutas para los otros niveles pre-escolar. Además, se puede apreciar que existen las hojas de cálculo para los demás meses del año.

Con todos estos datos ya ingresados en el archivo, este mismo calcula la cantidad de cada producto a comprar para el mes (ingredientes supermercado) o para la semana (ingredientes perecibles), de esta forma la administradora del jardín puede llevar un registro del stock disponible de cada producto y poder comprar el faltante de ingredientes. De esta manera, se optimizan los productos disponibles y así no se compra más de lo que se va a ocupar. La figura 2.3 muestra un ejemplo de cómo se ve el stock para el mes de Julio 2021.

En primera instancia, con los recursos y conocimientos disponibles es una buena solución para realizar el proceso de creación de minutas, sin embargo no es una solución óptima, ya que los tiempos para la creación de las minutas por mes son mucho mayores si se crea una plataforma tecnológica para la creación de minutas. Además, se puede observar en la figura 2.4 que es necesario definir los días de los meses de cada año, así como otros parámetros.

Figura 2.1: Hoja de cálculo de base de datos para minutas

[illegible]

18

Capítulo 3

Marco tecnológico

En la actualidad, se disponen de soluciones comerciales, enfocadas principalmente al rubro de la nutrición. Estas soluciones cumplen la función de calcular los nutrientes por plato; seleccionar comidas y menús de acuerdo a los límites de calorías, colesterol, sodio o cualquier otro nutriente.

3.1. DietPlan

DietPlan ¹, es un software de análisis nutricional exclusivo para nutricionistas profesionales, disponible para Windows y macOS. Este software permite analizar alimentos, comidas y menús; diseñar planes alimenticios o minutas para pacientes. Además, determina los requerimientos de nutrientes; selecciona las comidas y menús de acuerdo con el límite de calorías, colesterol, sodio, etc. DietPlan utiliza una base de datos preinstalada que incluye tablas de alimentos creada en Reino Unido, pero es posible utilizar tablas de alimentos diseñadas por el propio usuario, es decir, agregar un número ilimitado de sus propios alimentos. Cabe destacar de este software, la posibilidad de descargar los informes generados en formato PDF.

¹<https://www.foresoft.co.uk/index.html>

3.2. Food Processor

Food Processo², es un software de análisis nutricional de la empresa ESHA, combina una gran e investigada base de datos de alimentos e ingredientes con una interfaz ordenada y sencilla. Desde 1984, profesionales de la nutrición, restaurantes e instituciones de educación han utilizado esta herramienta para analizar menús, dietas, alimentos e incluso acondicionamiento físico de los clientes/pacientes. Una de las principales funciones a destacar de Food Processor es la posibilidad de la generación de informes, los cuales se pueden visualizar en la misma plataforma, además de tener la posibilidad de imprimirlos. Otra funcionalidad para destacar es el módulo de planificación de menús y software de nutrición. Este módulo es una característica especial para clínicas, hospitales, asilos de ancianos y comedores de instituciones escolares. En este módulo se pueden crear planes de menú a partir de alimentos y recetas de uso frecuente, analizar los menús y comidas en busca del valor nutricional, se puede crear plantilla de menú en torno a estándares nutricionales que coincidan con las necesidades dietéticas del organismo en que se está utilizando esta herramienta. Otra característica de este módulo es la posibilidad de organizar los alimentos por comida, es decir, desayuno, almuerzo y cena; en categorías tales como jugos, entradas, plato de fondo, entre otros.

Además, se encuentran los siguientes software de alimentación institucional que están enfocados a la alimentación colectiva, ya sea en empresas, colegios, universidades, entre otros.

3.3. Menutech

Menutech³, es una plataforma online para planificar y preparar menús variados y equilibrados. Esta aplicación es capaz de adaptarse a cualquier tipo de comedores de hoteles, restaurantes, centros médicos, centros educativos, entre otros. Ofrece la posibilidad de crear cartas y menús para distintos tipos de instituciones, además de poder

²<https://esha.com/products/food-processor/>

³<https://menutech.com/es>

ampliar el espacio de trabajo digital con módulos de gestión de inventarios, recetas y comandas. Las cartas y menús que genera la aplicación cuenta con distintos formatos, tales como: PDF, HTML, imprimibles, la posibilidad de compartir con un link o código QR. Es una aplicación alojada en la nube, intuitiva, fácil de usar y completamente adaptable a los procesos de trabajo en restaurantes, bares, cafeterías, entre otros. Cabe destacar, que Menutech está adaptada a los centros educativos, en los cuales se pueden crear menús diarios, semanales o mensuales, para así entregar una planificación o un informe de los menús a servir en cada día del mes. Además, de tener la posibilidad de crear menús personalizados basados en la necesidad de cada estudiante, por ejemplo, se tiene la posibilidad de confeccionar menús con opciones veganas. Los menús personalizados son enviados a cada estudiante que tiene esa opción de alimentación.

3.4. Nutrigreen

Nutrigreen [2], es una herramienta informática desarrollada por el equipo de trabajo integrado por Javiera García (Tesisista de magíster en Ciencias de la Ingeniería), Héctor Arraigada (Nutricionista) y Marcos Mella (Ingeniero Informático). Esta herramienta fue ganadora del concurso Despega Usach del año 2021, la cual, automatiza la elaboración de minutas para el Casino Central Usach. En primera instancia, esta herramienta entrega sugerencias de minutas en una tabla de Excel con la información nutricional por plato y menú, dirigido a concesionarias de casinos de alimentación, con el fin de que la planificación del almuerzo fuera automático, reduciendo tiempo y costo en la fabricación de minutas e impacto ambiental. La planificación que se entrega en la tabla de Excel puede ser mensual, trimestral y hasta semestral. Durante el primer semestre del año 2022 el equipo espera desarrollar un MVP (Minimum Viable Product, es un prototipo del producto a desarrollar con funciones mínimas que permite presentarlo a clientes o entidades potenciales y comprobar el interés del producto) el cual sería convertir el algoritmo con el modelo matemático que desarrollaron y la visualización de las sugerencias de minutas en un Excel a un software de servicio SaaS, el cual permite a

los usuarios conectarse a aplicaciones basadas en la nube a través de internet, además de conectar el software a un gestor de base de datos.

3.5. Minuta Virtual

MinutaVirtual ⁴, es una plataforma web para nutricionistas y caseros de alimentación colectiva, en la cual se pueden hacer cálculos de nutrientes, ingredientes, pedidos a bodega, lista de compras y cálculo de costos. Esta plataforma está desarrollada por la empresa NUTRYSOFT ubicada en Puerto Montt, Chile. Además, soluciona los requerimientos de los profesionales del rubro de la alimentación, especialmente en caseros que deben procesar un gran número de raciones al día. Esta plataforma cuenta con una base de datos con más de 600 alimentos que son proporcionados por el INTA (Instituto de Nutrición y Tecnología de los Alimentos, de la Universidad de Chile). Cuenta con un módulo de diseño de recetas en el cual de forma rápida y fácil se pueden generar cálculos de nutrientes y costos, además de poder ingresar los pasos de la preparación. También tiene la posibilidad de poder diseñar minutas de alimentación para un día completo con recetas para el desayuno, almuerzo, cena, entre otras comidas. Otra función, es entregar un archivo imprimible para poder tener en físico las minutas y recetas.

Finalmente, se puede encontrar distintas aplicaciones que tienen como función planificar la alimentación de un grupo pequeño de personas o de manera personal. Un ejemplo de esto es la aplicación Mealime.

Mealime ⁵, es una aplicación móvil desarrollada para smartphones con sistema operativo iOS y Android. En esta aplicación es posible planificar comidas, recetas y tener de manera adicional una lista de compras. La aplicación organiza la lista de compras para el menú de la semana, de tal manera que así se generan menos desperdicios. Contiene recetas muy bien explicadas, lo cual hace que optimice el tiempo a la hora de cocinar.

⁴<https://nutrifysoft.com/software-minutavirtual/>

⁵<https://www.mealime.com/>

El cliente descarta las alternativas anteriores, ya que desea una plataforma web personalizada a sus requerimientos. Al ser propietario de este sistema es posible ofrecer esta plataforma a otros jardines infantiles para obtener un beneficio económico.

Otra alternativa para la creación de minutas es el desarrollo de una plataforma web. En la siguiente sección, se presenta las distintas tecnologías para poder desarrollar esta plataforma.

3.6. Tecnologías

Las tecnologías en desarrollo web avanzan de manera rápida y es fundamental estar al día con las mejores. Las diferentes opciones de tecnologías permiten crear interfaces de usuario y establecer la comunicación con el servidor. Es necesario hacer un diferenciación entre los elementos que componen cualquier desarrollo web.

Primeramente, está el frontend, que es la parte visible de la plataforma a desarrollar, donde los usuarios pueden interactuar y acceder al contenido. Por otro lado, se encuentra el backend, el cual es el encargado de la lógica y permite que la interfaz funcione, además de procesar la información y conectarse con el servidor [3].

Para poder gestionar estos dos elementos, es necesario una series de lenguajes y frameworks.

3.6.1. Lenguaje de programación

Un lenguaje de programación, es un lenguaje de computadora utilizado para desarrollar programas de software, aplicaciones, páginas webs, scripts u otros conjuntos de instrucciones para que sean ejecutadas por los ordenadores. En la actualidad existen más de 675 lenguajes de programación [4], por lo que es necesario elegir algún lenguaje de programación popular, donde exista una gran comunidad y documentación para ser consultada. Entre los lenguajes de programación disponibles, podemos encontrar: Python, PHP, Java, C#, JavaScript, entre otros. Dentro de todos estos lenguajes, JavaS-

cript es el lenguaje más usado para desarrollar frontend o backend. Según la encuesta de Stack Overflow del año 2022 [5], JavaScript es el lenguaje más usado por décimo año consecutivo. Por otro lado, su curva de aprendizaje es una de las más sencillas a comparación de lenguajes como Java o PHP, que tienen un desempeño similar.

3.6.2. Tecnologías para Frontend

Un framework de desarrollo, es una serie de herramientas y librerías de código pre-escrito que facilitan a los desarrolladores las tareas cotidianas o de uso común para evitar ejecutarlas desde cero. Los frameworks, dan un marco de trabajo para programar en un lenguaje y las librerías se encargan de solucionar problemas produciendo un código más legible.

En este análisis de tecnologías, solo se detallarán las que son más utilizadas actualmente por los desarrolladores. Las 3 tecnologías que se presentan a continuación tienen algunas similitudes, como su lenguaje de programación (JavaScript) y su base en componentes esto permite modularizar la aplicación, lo cual separa, organiza, estructura y reutiliza código.

Los frameworks a analizar se presentan en el orden de popularidad según la encuesta de Stack Overflow del año 2022 [5].

React.js

Es una biblioteca de JavaScript de código abierto para construir interfaces de usuario, desarrollada por Meta Open Source ⁶. Está enfocada en la creación de componentes interactivos y reutilizables. Éste es un paradigma llamado programación orientada a componentes, en donde cada componente es una pieza con la que el usuario puede interactuar. Estos componentes, se crean usando la sintaxis JSX, la cual permite escribir HTML dentro de objetos JavaScript. Actualmente, muchas empresas de primer nivel utilizan este framework para el desarrollo de sus aplicaciones, tales como: Instagram,

⁶<https://opensource.fb.com/>

Uber, Netflix, Twitter o Paypal.

Ventajas

- Amplia comunidad: al estar basado en JavaScript, cuenta con una amplia comunidad que dispone de un gran número de librerías externas.
- Proporciona componentes reutilizables: otorga a los desarrolladores la autoridad para reutilizar los componentes creados para alguna otra aplicación que tenga la misma funcionalidad. De esta manera, se reduce el esfuerzo y tiempo de desarrollo.
- Flexible: cuando se utiliza un framework, viene con funcionalidades integradas, tales como: testing, routing, validación de formularios, entre otros. Con React se pueden elegir entre varias librerías para soportar la función que se necesite.

Desventajas

- Ausencia de documentación oficial: la documentación existente es creada por los mismos desarrolladores que crean bibliotecas para React.js. Por lo cual, es difícil encontrar un enfoque común entre ellos.
- Sintaxis JSX: algunos desarrolladores encuentran que es una desventaja de React.js, ya que el código en JSX es demasiado complejo y difícil de entender.
- Ritmo de desarrollo: este framework está en constante evolución y cambio. Por lo que es necesario estudiar y reaprender constantemente.

Angular

Angular es un framework de código abierto. Su principal objetivo es desarrollar aplicaciones de una sola página. Es mantenido por Google, es decir, se encarga de darle soporte y ofrecer actualizaciones de mejora.

Ventajas

- Documentación oficial: la documentación de Angular está cuidadosamente escrita, además de presentar una gran variedad de ejemplos de código para tener mayor claridad en el proceso y así permitir que los desarrolladores encuentren soluciones rápidas a cualquier problema mientras crean una aplicación.
- Soporte de Google: cuando se definió este framework, se destacó que es mantenido por Google, por lo cual, se puede estar seguro de que se recibirá soporte a cualquier duda o problema al momento de desarrollar.
- Implementación de la arquitectura MVC: Angular implementa la arquitectura MVC, lo cual hace posible aislar la lógica de la aplicación de capa de la interfaz de usuario. El controlador recibe todas las solicitudes de la aplicación y opera con el modelo para preparar los datos que necesita la vista. La vista utiliza los datos preparados por el controlador y muestra una respuesta presentable en la vista.

Desventajas

- Curva de aprendizaje pronunciada: la dificultad para aprender Angular es alta comparada con otros frameworks.
- Fallos de rendimiento: las aplicaciones desarrolladas en Angular son dinámicas. Estas aplicaciones dinámicas carecen de rendimiento provocando algunos retrasos y fallos.
- Detallado y complejo: aunque está basado en componentes, como principal beneficio, la forma en cómo se administran los componentes es demasiado complicada. Por ejemplo, es posible que se necesiten hasta 5 archivos para un solo componente.

Vue.js

Es un framework de código abierto de Javascript, creado por el ex trabajador de Google Evan You, que ha conseguido posicionarse entre los más usados por la comunidad junto a Angular y React.js. Es considerado un framework progresivo, debido a que puede ser usado tanto para tareas básicas, como para tareas complejas. Un ejemplo de tareas complejas es la creación de una SPA (Single-page application) completa.

Ventajas

- Curva de aprendizaje: es uno de los frameworks más sencillos para comenzar a desarrollar, ya que es considerado "developer friendly". Tiene una excelente documentación y una amplia comunidad que proporciona muchos ejemplos.
- Tamaño pequeño: puede ocupar a lo más 20 KB, manteniendo su velocidad y flexibilidad permitiendo alcanzar un mejor rendimiento en comparación con otros frameworks.
- Componente de un solo archivo: utiliza una arquitectura basada en componentes, separando grandes trozos de código en componentes más pequeños. Cada componente está escrito con HTML, CSS, JavaScript, lo que favorece la claridad y simplicidad en su código.

Desventajas

- Falta de documentación completa en inglés: las empresas que ocupan este framework principalmente son chinas (Xiaomi y AliBaba), por lo que ha motivado a los desarrolladores de esos países a convertirse en expertos y a publicar mucha información en su idioma.
- Framework nuevo: lleva pocos años en el mercado, por esa razón, hay pocos desarrolladores expertos, lo que conlleva a ser uno de los menos utilizados en el mercado laboral y genera poco interés a la hora de aprenderlo.

- Flexibilidad de código excesiva: la flexibilidad puede generar irregularidades y errores en el código. La mayoría de los retrasos en el código se deben a una flexibilidad excesiva, cuando se pueden aplicar varios enfoques de programación diferentes simultáneamente dentro del mismo equipo de desarrolladores.

3.6.3. Tecnologías para Backend

Para el análisis de tecnologías framework para backend, se detallarán las preferidas entre los desarrolladores. Los frameworks analizados se presentan en el orden de popularidad según la encuesta de Stack Overflow del 2022 [5].

Express.js

Es un framework JavaScript que utiliza Node.js para crear el lado del servidor de aplicaciones web. Node.js es un entorno de ejecución de JavaScript backend de código abierto. Express destaca por contener métodos de utilidad HTTP, además de poseer gran cantidad de middleware. Un middleware es una función que se puede ejecutar antes o después del manejo de una ruta. Esta función tiene acceso al objeto Request, Response y la función next(). Las funciones middleware suelen ser utilizadas como mecanismo para verificar niveles de acceso antes de entrar en una ruta, manejo de errores, validación de datos, entre otros.

Ventajas

- Desarrollo rápido de aplicaciones: permite usar el mismo lenguaje JavaScript tanto en backend como en frontend. Brinda a los desarrolladores la oportunidad de convertirse en fullstack. Por lo tanto, el proceso de desarrollo es mucho más rápido y sencillo, ya que una sola persona puede gestionar toda una aplicación.
- Comunidad de código abierto: posee una comunidad de código abierto, por lo que el código siempre se revisa y mejora.

- Fácil integración de servicios y middleware de terceros: como es un framework minimalista, el equipo de desarrollo de Express ha creado paquetes de middleware para resolver diferentes problemas de desarrollo, entre estos: parámetros URL, sesiones, encabezados de seguridad, entre otros.

Desventajas

- Muchas líneas de código: al ser un framework liviano, se requieren más líneas de código para crear una aplicación en relación con otros framework.
- No es robusto en seguridad: express no menciona cómo se implementa la seguridad en una aplicación. El nivel de seguridad dependerá del tipo de aplicación y del desarrollador.

ASP.NET Core

Es un framework gratuito de código abierto desarrollado por Microsoft. Proporciona funciones que permiten crear el backend para aplicaciones web modernas, así como API web. El lenguaje de programación que se utiliza para el desarrollo de ASP.NET Core es C# o cualquier otro lenguaje de programación basado en .NET. Es considerado como uno de los framework de backend de mejor rendimiento.

Ventajas

- Permite la separación de intereses: sigue la estructura MVC, que permite separar la entrada, el proceso y la salida de la aplicación.
- Caja de herramientas de primera clase: este framework viene con una caja amplia de herramientas a través de su entorno de desarrollo integrado Visual Studio. Esto ayuda al desarrollador a crear aplicaciones de forma más rápida.
- Seguridad: se pueden desarrollar aplicaciones seguras a través de la autenticación de Windows incorporada y las características de configuración por aplicación.

Desventajas

- **Coste:** en comparación con las alternativas de código abierto, ASP.NET Core es más costoso, ya que tiene gastos como licencias de SQL Server, licencias de Visual Studio, licencias de servidor de Windows, entre otros.
- **Escaso soporte comunitario:** aunque ASP.NET cuenta con actualizaciones de seguridad constantes, no tiene foros de colaboración desinteresada lo suficientemente amplios.
- **Elevado consumo de recursos:** el uso de ASP.NET necesita variados recursos para su correcto funcionamiento y compilación de recursos programados, ello significa más memoria ram y velocidad de procesamiento del CPU.

Django

Es un framework de aplicaciones web gratuito y de código abierto escrito en Python. Se trata de un lenguaje que se emplea, sobre todo, en la programación para las distintas aplicaciones de análisis de datos e IA, pero que también es importante para la programación web. Cuenta con una comunidad amplia que comparte recursos constantemente. Django es una herramienta que se puede usar tanto para el desarrollo fullstack de aplicaciones y páginas web, como para el desarrollo de servidores.

Ventajas

- **Portabilidad:** este framework se puede usar en cualquier sistema y plataforma, además está respaldado por muchos de los proveedores de hosting.
- **Múltiples paquetes:** cuenta con variedad de paquetes para resolver la mayoría de los problemas comunes. Además, estos paquetes son supervisados y mejorados por la comunidad, lo que garantiza una calidad de la aplicación.

- Amplia documentación: la documentación tiene una amplia gama de casos prácticos y documentos en línea. Los desarrolladores pueden buscar documentos en línea que buscan soluciones cuando se encuentran en problemas.

Desventajas

- Es lento y maneja peticiones de una en una: Python es un lenguaje interpretado que se hizo para ser simple, no necesariamente rápido. Por lo cual, Django no destaca por su velocidad a la hora de ejecutarse. Además, es incapaz de permitir que los procesos individuales manejen múltiples peticiones simultáneamente.
- Naturaleza monolítica: esto puede ser una buena característica para algunos programadores, pero para la mayoría es un inconveniente. Django tiene un cierto conjunto de archivos y variables predefinidas que los programadores necesitan aprender antes de crear cualquier proyecto. El framework tiene una determinada forma de realizar las tareas.
- Curva de aprendizaje empinada: una de las principales desventajas es que Django tiene una curva de aprendizaje pronunciada. Viene con muchas características y configuraciones que los desarrolladores no pueden entender fácilmente. Aunque el framework es antiguo y evoluciona lentamente, sigue siendo uno de los más difíciles de dominar.

3.6.4. Base de datos

Para poder guardar y acceder a los datos ingresados por los usuarios de la plataforma, es necesario tener una estructura de base de datos donde sea posible recopilar, organizar y relacionar los datos entre sí, para que se pueda hacer una rápida búsqueda a través del navegador web. Existen dos tipos de base de datos: relacionales y no relacionales. Existe una variedad de sistemas para poder administrar estos tipos de base de datos (MySQL, MariaDB, MongoDB, Redis, entre otros). Cabe mencionar que en esta

sección solo se presentan los tipos de base de datos y no los administradores de base de datos, ya que éstos últimos dependen de las tecnologías que se usen en el desarrollo.

No relacional

Una base de datos no relacional es aquella que no usa el esquema tabular de filas y columnas que se encuentra en la mayoría de sistemas de base de datos tradicionales. Usan un modelo de almacenamiento que está optimizado para los requisitos específicos del tipo de datos que se almacena. Los datos que pueden almacenar van desde pares de clave/valor, documentos JSON hasta grafos [6].

Características

- Las bases de datos no relacionales no tienen un identificador que sirva de relación entre unas tablas y otras.
- La información no se almacena en tabla, sino en documentos.
- Muy útiles para gestionar y organizar información no estructurada o cuando no se conocen bien los datos a almacenar.
- Son bases de datos con alto nivel de escalabilidad y están diseñadas para soportar grandes volúmenes de datos.
- Es un sistema relativamente nuevo, por lo que aún no está debidamente estandarizado.

Relacional

Una base de datos relacional, es un conjunto de tablas formadas por filas y columnas, donde cada registro, es decir, cada fila, tiene un ID único denominado clave y las columnas contienen los atributos de los datos. Además, emplea el modelo relacional

para representar y crear uniones entre los distintos datos, de forma que se pueden consultar y actualizar usando el lenguaje SQL [7].

Características

- Una de las principales características es que evita la duplicidad de registros y a su vez garantiza la integridad referencial, es decir, que si se elimina uno de los registros, la integridad de los registros restantes no será afectada.
- Gracias a las claves se puede acceder de forma sencilla a la información y recuperarla en cualquier momento.
- Las relaciones que se pueden establecer entre distintas tablas permiten juntar y extraer datos de ellas como si fuera de una sola. Estas relaciones se llevan a cabo, usando lo que se conoce, como claves primarias y claves foráneas.
- Ya que las bases de datos relacionales acaban segmentándose en diferentes tablas, esto provoca un rendimiento negativo a la hora de hacer consultas y obtener la información deseada.

Capítulo 4

Desarrollo de la plataforma

Como se explicó en el capítulo 2 la administradora del jardín lleva el control del stock, número de matrículas y es quien elabora las minutas alimenticias en un archivo .xlsx. Por ser un proceso lento y difícil para ella, es que ha decidido dejar de utilizar este archivo. Por lo tanto, no se ha llevado un control sobre el inventario de los productos perecibles y supermercado que requiere la cocina para producir las distintas colaciones.

Por estas razones, es que se propone crear una aplicación web con un sistema de usuario que permita de forma rápida y sencilla obtener el listado de productos a comprar, tanto para semana (para productos perecibles) como para el mes (productos comprados en supermercado). Además, poder calendarizar las minutas alimenticias por mes o año para los distintos niveles del establecimiento.

En el presente capítulo, se presenta en detalle lo expuesto en la sección 1.2, iniciando con la definición de los requerimientos y finalizando con el desarrollo de la plataforma web. La implementación de la plataforma en un servidor web se analizará en el capítulo 5 de resultados.

4.1. Requerimientos de la plataforma

Antes de iniciar el desarrollo de la plataforma, es necesario definir los requerimientos necesarios para que el sistema funcione correctamente. Estos requerimientos pueden

ser funcionales o no funcionales, lo cuales se desprenden del requisito principal de la plataforma.

Los requerimientos son levantados a través de reuniones con la contra-parte del proyecto, en este caso, la administradora del jardín; y a través del análisis del proceso de creación de minutas presentado en la sección 2.2.

El requisito principal de la plataforma es la creación, actualización y visualización de minutas alimenticias.

4.1.1. Requisitos funcionales

- **Sistema de usuarios y roles.**

Para ingresar a la aplicación se debe entrar con el nombre de usuario y contraseña. Los usuarios tienen un rol único dentro de la aplicación, los cuales pueden ser administrador, nutricionista y manipuladora de alimentos. El administrador tiene la facultad de crear nuevos usuarios dentro de la aplicación, crear nuevos ingredientes, nuevos platos, nuevas variables, actualizar stock, crear matrículas, generar un archivo pdf con el calendario de las minutas y lista de compras semanal o mensual, entre otras funcionalidades. Es decir, el administrador puede realizar todas las funciones dentro de la aplicación. La nutricionista, tiene la función de crear nuevas recetas, es decir, elegir los ingredientes y su cantidad de cada plato dependiendo del nivel pre-escolar. Además, puede crear las minutas de manera diaria, mensual o anual. Finalmente, la manipuladora de alimentos solo tiene acceso a un calendario con las minutas separadas por nivel escolar. En este calendario, ella puede encontrar las medidas de cada ingrediente para cada plato y la cantidad de niños que están asociados a ese plato.

- **Multisucursal.**

Natgio solo tiene un establecimiento en Quilpué, pero existe la posibilidad que en los próximos años se abran nuevos establecimientos en la región de Valparaíso u otras regiones de Chile. Por lo cual, es imprescindible la creación de nuevas

sucursales dentro de la aplicación.

- **Niveles educativos.**

Existe la posibilidad, que en un futuro, Natgio tenga más niveles educativos, los cuales tendrán sus respectivas recetas y minutas.

- **Personalización de variables del sistema.**

Es necesario la posibilidad de crear otros tipos de colaciones, ingredientes y recetas.

- **Listado de ingredientes.**

Dentro de la aplicación es necesario poder obtener un listado de todos los ingredientes que se van a utilizar en las minutas, de manera semanal o mensual. Este listado debe de estar asociado a su stock actual para poder mostrar la cantidad exacta a comprar de cada ingrediente. Por disposición de la contra-parte, no es necesario llevar un historial de stock, por lo cual, se debe actualizar antes de fin de mes o antes que termine la semana.

- **Una misma receta para distintos tipos de niveles.**

Al crear las recetas, es necesario definir el nivel escolar y el tipo de colación a la cual estará asociada la receta. Las recetas varían dependiendo de su nivel pre-escolar y/o tipo de colación.

- **Configuración mensual de matrículas.**

Mensualmente se debe ingresar la cantidad de niños (matrículas) por nivel pre-escolar y tipo de colación, para poder obtener la lista de compra de cada ingrediente.

- **Generar informes en PDF.**

Las minutas puedan ser descargadas en formato pdf de forma mensual (estilo calendario) y por nivel escolar, para que puedan ser enviadas a los apoderados de cada niño del jardín. Otro informe a descargar en formato pdf es la lista de

compras. Además, se debe generar un documento donde la manipuladora de alimentos pueda obtener la receta con la cantidad de ingredientes y la cantidad de niño.

4.1.2. Requerimientos no funcionales

- **Aplicación disponible 24/7.**

La plataforma debe estar operativa las 24 horas del día los 7 días de la semana. El sistema tiene que tener una disponibilidad superior al 95 %, de modo que el acceso a los datos sea de forma continua.

- **Interfaz amigable y conocida para el usuario**

Se requiere que el sistema sea cómodo para el usuario al momento de su uso. Se utiliza un estilo similar al Calendario de Google [8] para la creación y visualización de las minutas.

4.2. Tecnologías a ocupar

En el capítulo 3 se estudian algunas tecnologías con sus ventajas y desventajas, cabe mencionar que cada una de las tecnologías es suficiente y puede cumplir el desarrollo de la plataforma para la creación de minutas. Por lo cual, la elección está basada en el conocimiento previo del lenguaje y del framework, curva de aprendizaje, documentación disponible y rendimiento.

Los framework, ya sea para frontend o backend, usan los siguiente lenguajes de programación:

Framework	Lenguaje
React.js	JavaScript
Angular	JavaScript
Vue.js	JavaScript

Cuadro 4.1: Lenguajes de programación para frameworks de frontend

Framework	Lenguaje
Express.js	JavaScript
ASP.NET Core	C#
Django	Python

Cuadro 4.2: Lenguajes de programación para frameworks de backend

Las siguientes tablas muestran un resumen con el nivel cualitativo para cada uno de los ítems de elección de frameworks.

	Express	ASP.NET Core	Django
Conocimiento previo del lenguaje	Alto	Bajo	Alto
Conocimiento previo del framework	Alto	Nulo	Medio
Curva de aprendizaje	Bajo	Medio	Medio
Documentación disponible	Alto	Alto	Medio
Rendimiento y flexibilidad	Alto	Alto	Alto

Cuadro 4.3: Resumen calificaciones para framework backend

	React.js	Angular	Vue.js
Conocimiento previo del lenguaje	Alto	Alto	Alto
Conocimiento previo del framework	Medio	Bajo	Alto
Curva de aprendizaje	Media	Alta	Baja
Documentación disponible	Alto	Alto	Alto
Rendimiento y flexibilidad	Alto	Medio	Alto

Cuadro 4.4: Resumen calificaciones para framework frontend

4.2.1. Base de datos

Con el archivo facilitado por la administración del jardín, es posible saber cómo se van construyendo las minutas alimenticias y las relaciones que existen entre cada elemento, ya sea ingredientes, niveles, recetas, entre otros. Tomando en cuenta los requerimientos de la plataforma y este archivo, es que se decide usar una base de datos relacional. El archivo facilitado por la administración cuenta con hojas de cálculo que funciona como base de datos. Por lo que es necesario modificar estas tablas para crear las relaciones entre los elementos que componen la plataforma.

4.2.2. Frameworks

Todo el análisis realizado en el capítulo 3 fue para elegir las mejores tecnologías que se adapten a los requerimientos de la plataforma. Cabe destacar que además de un análisis técnico se realiza también un análisis de conocimiento previo de cada tecnología para determinar el tiempo necesario que se empleará para aprenderla. Se eligen los framework de desarrollo en los cual se tiene un mayor conocimiento previo, ya que los tiempos para desarrollar son limitados y es necesario enfocarse en el desarrollo y no en aprender una tecnología nueva.

Es por eso que se define el siguiente stack (conjunto de tecnologías): para frontend (lado del cliente) se utiliza el framework Quasar, el cual está basado en Vue.js; para backend se utiliza (lado del servidor) una interfaz de programación de aplicaciones (API) en Express.js, de tal forma que la API se conecta con la base de datos y con el cliente.

Cada una de estas tecnologías ocupa el lenguaje de programación JavaScript, convirtiéndolo en una ventaja debido a que todo el desarrollo de la plataforma está en el mismo lenguaje.

La figura 4.1 muestra la división de la plataforma (en frontend y backend), con sus respectivas tecnologías utilizadas.

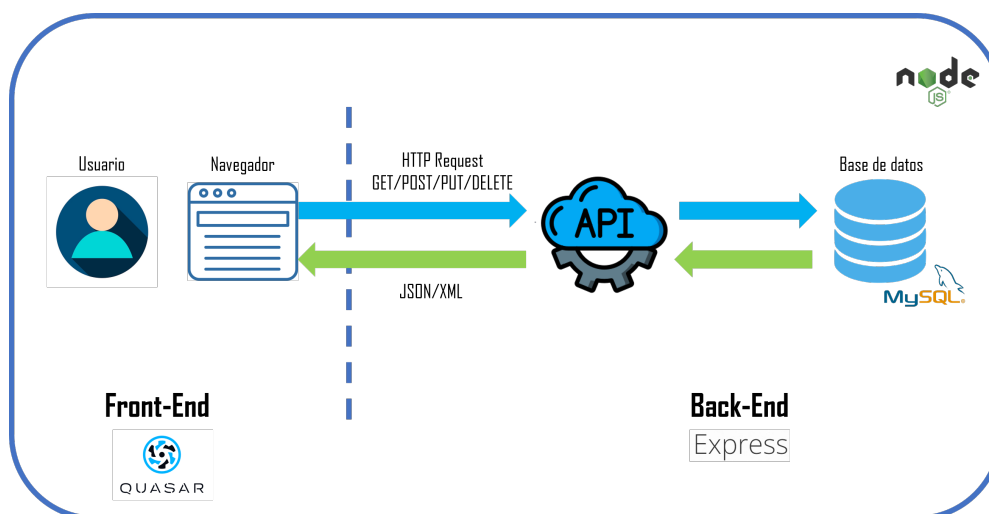


Figura 4.1: Diagrama de arquitectura de la plataforma

- **MySQL**: es un sistema de gestión de bases de datos relacionales, basado en el lenguaje de consulta estructurado (SQL).
- **Express**: parte de backend, donde se levantan las APIs que se encargan de recibir las peticiones HTTP y devolver los datos solicitados.
- **Quasar** ¹: Framework de desarrollo basado en Vue.js para realizar la interfaz web. Es un framework de código abierto y ha demostrado ser el más fácil de usar para crear aplicaciones de una sola página(SPA), proyectos de renderizado del lado del servidor(SSR), aplicaciones web progresivas(PWA), así como aplicaciones móviles y de escritorio. Gracias a todos los componentes y herramientas que ofrece Quasar listas para usar, los desarrolladores no necesitan bibliotecas de terceros como Bootstrap o Hammer.js.
- **Node**: Es el entorno de ejecución multiplataforma.

4.3. Base de datos de la aplicación

A partir de los requerimientos de la plataforma y del archivo que fue entregado por la administración, es que se logra diseñar la base de datos relacional.

Para poder empezar a diseñar la base de datos es que se ocupa el diagrama entidad - relación, el cual permite ilustrar cómo las “entidades”(personas, objetos o conceptos), se relacionan entre sí dentro de un sistema. Es decir, cómo los usuarios, recetas, minutas, sucursales, entre otros, se relacionan entre sí en el sistema a desarrollar para la creación de minutas.

En la figura 4.2 se muestra el diagrama de entidad - relación de la base de datos. Se puede observar las distintas entidades que existen en la plataforma, además de cómo se relacionan entre sí a través de la cardinalidad en notación pata de gallo.

¹<https://quasar.dev/>

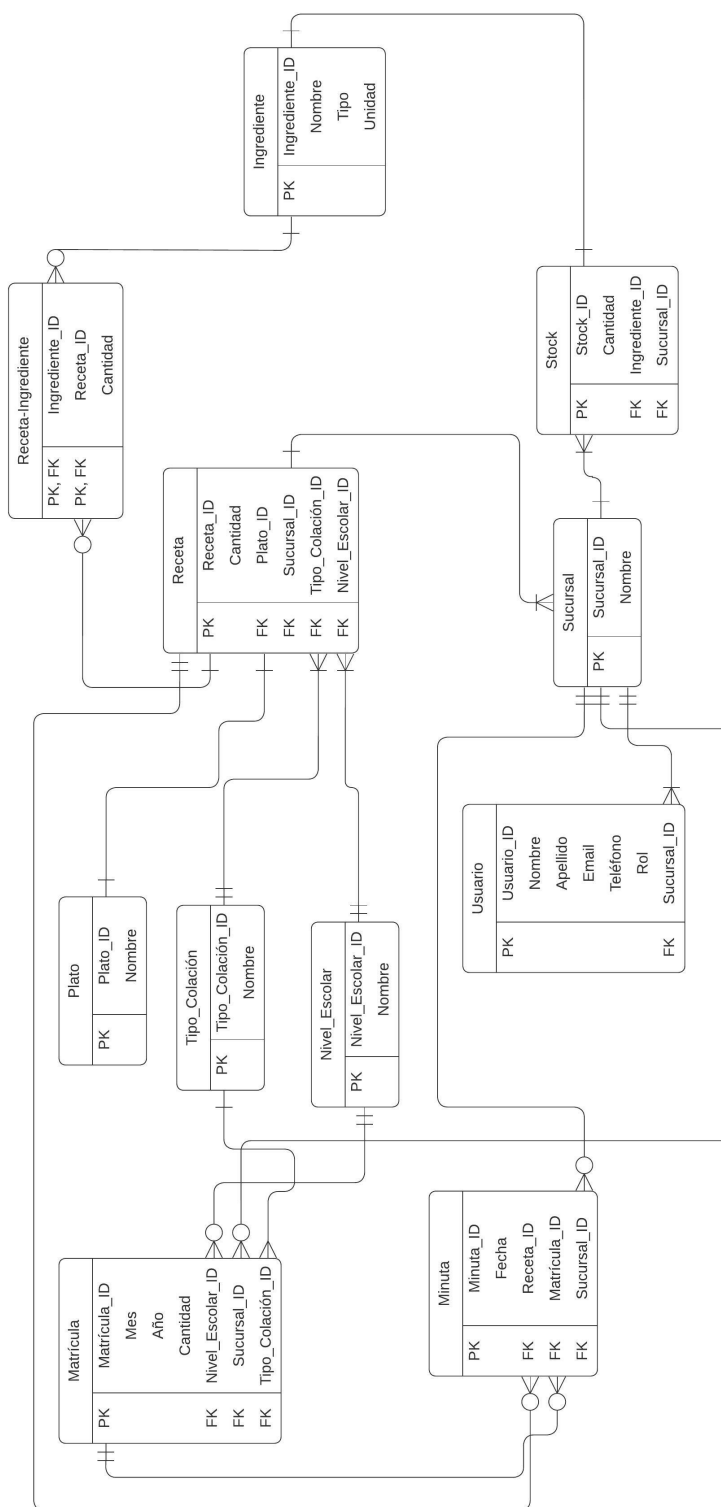


Figura 4.2: Diagrama Entidad - Relación de la base de datos, donde la entidad usuarios interactúa con las demás entidades que participan en la creación de minutas.

4.4. Backend de la aplicación

Tal como muestra la arquitectura de la plataforma en la figura 4.1, en la capa de negocios se desarrolla una API en Express.js, la cual se conecta a una base de datos relacional a través de MySQL. Esta API se encarga de recibir las peticiones HTTP de parte del cliente y hacer las consultas correspondientes a la base de datos, es decir, hace posible la comunicación entre el cliente y la base de datos.

Las siguiente tablas muestran cómo están estructurados los endpoints de la API para las minutas, ingredientes y stock. Los demás endpoints se pueden ver en el anexo C.

Ruta/ Descripción	Método	Header
<i>host/api/minutas</i> Listado de minutas	GET	Content-Type: application/json auth-token: token
<i>host/api/minutas</i> Registrar una nueva minuta	POST	Content-Type: application/json auth-token: token
<i>host/api/minutas/copiar</i> Copiar minutas	POST	Content-Type: application/json auth-token: token
<i>host/api/minutas/:minuta_id</i> Editar una minuta	PUT	Content-Type: application/json auth-token: token
<i>host/api/minutas/:minuta_id</i> Traer la información de una minuta	GET	Content-Type: application/json auth-token: token
<i>host/api/minutas/:inicio_mes/:fin_mes</i> Traer minutas de un mes en específico	GET	Content-Type: application/json auth-token: token
<i>host/api/minutas/:minuta_id</i> Eliminar una minuta	DELETE	Content-Type: application/json auth-token: token

Cuadro 4.5: Tabla de endpoints de minutas

Ruta/ Descripción	Método	Header
<i>host/api/ingredientes</i> Listado de ingredientes	GET	Content-Type: application/json auth-token: token
<i>host/api/ingredientes</i> Registrar un nuevo ingrediente	POST	Content-Type: application/json auth-token: token
<i>host/api/ingredientes/:ingrediente_id</i> Editar un ingrediente	PUT	Content-Type: application/json auth-token: token
<i>host/api/ingredientes/:ingrediente_id</i> Traer la información de un plato	GET	Content-Type: application/json auth-token: token
<i>host/api/ingredientes/:ingrediente_id</i> Eliminar un ingrediente	DELETE	Content-Type: application/json auth-token: token

Cuadro 4.6: Tabla de endpoints de ingredientes

Ruta/ Descripción	Método	Header
<i>host/api/stock</i> Listado de stock	GET	Content-Type: application/json auth-token: token
<i>host/api/stock</i> Registrar un nuevo stock	POST	Content-Type: application/json auth-token: token
<i>host/api/stock/:stock_id</i> Editar un ingrediente	PUT	Content-Type: application/json auth-token: token
<i>host/api/stock/:stock_id</i> Traer la información de un stock	GET	Content-Type: application/json auth-token: token
<i>host/api/stock/:stock_id</i> Eliminar un stock	DELETE	Content-Type: application/json auth-token: token
<i>host/api/stock/listado-de-compra</i> Traer listado de compra	GET	Content-Type: application/json auth-token: token

Cuadro 4.7: Tabla de endpoints de stock

Cada uno de los endpoints contiene:

- **Ruta:** dirigen las solicitudes entrantes de la API a los recursos del backend. Además, pueden contener parámetros de consulta para obtener recursos más específicos.
- **Método:** se ocupan 4 tipos de métodos GET, POST, DELETE y PUT.
- **Header:** metadatos adicionales que se mandan a la API para ayudar al servidor a comprender qué tipo de solicitud se está mandando. En los endpoints se ocupa *auth-token* para agregar el token de usuario y *Content-Type: application/json* para indicar que el formato en que serán retornados los recursos sea en JSON. Este formato sirve para estructurar datos en forma de texto y permite el intercambio de información entre aplicaciones de manera sencilla, liviana y rápida, el cual puede ser utilizado en cualquier lenguaje de programación [9].
- **Body:** donde son enviados los recursos hacia el backend. Este elemento es enviado en formato JSON. Cabe mencionar que solo se envía un body cuando el método del endpoint es POST o PUT.

4.5. Frontend de la aplicación

Para lograr una comunicación entre el usuario y la API, es necesario el desarrollo del frontend de la aplicación. Las siguientes sub-secciones presentan los componentes que existen en cada vista, como se realiza el llamado hacia la API y el diagrama de flujo. Para saber los detalles de las demás vistas que no son presentadas en esta sección consultar el anexo D.

4.5.1. Login de usuarios

Para poder ingresar a la plataforma, es necesario autenticarse con el correo y contraseña. Como se mostró en los requisitos funcionales en la sub-sección 4.1.1, existen tres tipos de usuarios: administrador, nutricionista y manipulador de alimentos, los cuales pueden realizar distintas acciones dentro de la plataforma.

Luego de ingresar el correo y contraseña del usuario, la biblioteca Fetch API ², es la encargada de realizar el *HTTP Request* hacia el backend. Esto retorna un *Response* con la respuesta de la API.

Para iniciar sesión se realiza un HTTP Request del tipo POST hacia la ruta

host/api/users/login

En el headers del Request se indica que el formato de entrada de los datos es de tipo JSON, por lo cual, se agrega *Content-Type*: 'application/json'. En el body van las credenciales del usuario en formato JSON. Cabe destacar, que las variables se envían cifradas, para evitar el robo de estos datos.

Al momento de llegar el HTTP Request a la API, ésta hace las consultas necesarias a la base de datos para encontrar al usuario que quiere ingresar a la plataforma.

El Response que llega al frontend puede ser de dos formas:

- Credenciales Incorrectas: la API a través del Response, manda el código de es-

²https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

tado 400 Bad Request, mostrando que no ha podido encontrar al usuario en la base de datos. Seguido a esto, se muestra una notificación a través de la biblioteca Notify ³ de Quasar, indicando que el usuario o la contraseña son incorrectos. Además, la API no hace envío del token de autenticación; mecanismo de seguridad para el consumo de la API.

- **Credenciales Correctas:** La API a través del Response, manda el Código de estado 200 OK, el cual indica que las credenciales son correctas. Antes de iniciar sesión, se analiza el estado del usuario (1 = activo o 0 = desactivado). Si el usuario está desactivo no puede ingresar a la plataforma (indicando un mensaje con el error) y si está activo se procede a enviar el token de autenticación. Se almacena el token (ver anexo A) en el store de la plataforma, a través de la biblioteca Vuex (ver anexo B) y en el local Storage, también se almacenan en el store los datos del usuario para identificar si es un administrador, nutricionista o manipulador de alimentos. Dependiendo del rol del usuario, se redirigirá a la vista principal de la plataforma o a la de selección de sucursal.

El flujo de la vista de login, se puede observar en la siguiente figura.

³<https://quasar.dev/quasar-plugins/notify>

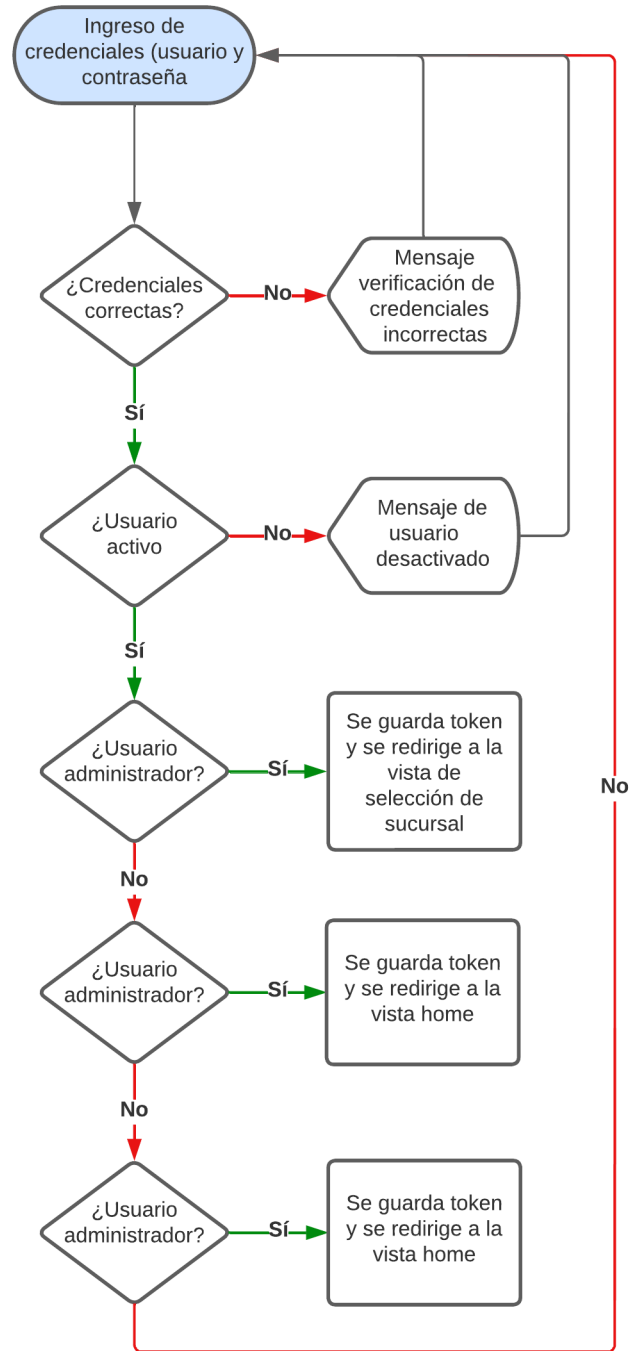


Figura 4.3: Diagrama de flujo de la vista login.

4.5.2. Vista Home

La vista Home, es la vista principal de la plataforma. Esta vista cuenta con varios componentes o acciones (ver anexo D.2). En esta subsección se presentan los siguiente:

Guardar minutas en PDF

Esta acción guarda en estilo tabla y en formato PDF las minutas de todos los niveles en un mes específico. Este archivo PDF, es impreso y entregado a la manipuladora de alimentos de la sucursal. Al ser accionado este botón, se realiza un Request de tipo GET a la URL

host/api/minutas?filtros=sucursalId|inicioMes|finMes

Se agrega un params de filtros para que la respuesta contenga todas las minutas de la sucursal en un mes específico. Al llegar el Response con el array de minutas, a través de la librería *jspdf* y su método *autotable*, es posible crear un archivo pdf con tablas de datos de las minutas. Las columnas de las tablas que se encuentran en el archivo, son: fecha, nivel, plato, ingredientes y medidas por porción. Cabe mencionar, que si el array de minutas que llega en el Response está vacío, aparecerá una notificación con un mensaje de error. Luego, a través del método *save* de la misma biblioteca, es posible descargar el archivo.

La siguiente figura muestra el diagrama flujo de esta acción.

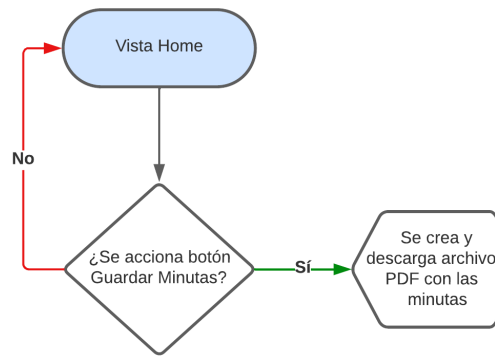


Figura 4.4: Diagrama de flujo guardar minutos en PDF

Guardar calendario en PDF

Esta acción, guarda en un archivo PDF el calendario del nivel que está en la vista. Esto es posible agregando la imagen del calendario que se obtiene con la biblioteca *domtoimage* y su método *toPng*. Este método realiza un screenshot al componente de calendario. Luego, se crea un archivo PDF con la biblioteca *jspdf* y se adjunta el screenshot con el método *addImage* y se descarga con el método *save*. Si no hay minutos en el calendario, se muestra un mensaje de error.

La siguiente figura, muestra el diagrama flujo de esta acción.

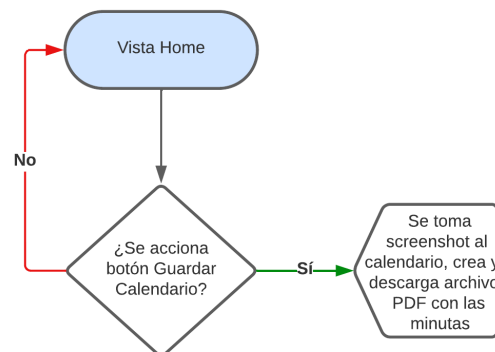


Figura 4.5: Diagrama de flujo guardar calendario en PDF

4.5.3. Menú de Navegación

Para poder acceder a las distintas vistas de la plataforma, se hace a través de un menú de navegación con varias opciones (ver anexo D.3). En esta subsección se presentan las siguientes:

Variables institucionales

Al accionar el botón Variables institucionales, éste redirige hacia la vista donde se pueden crear y listar las variables de sucursales y niveles.

En esta vista se cargan 4 componentes:

1. Formulario creación de sucursal: Este componente es el encargado de crear una nueva sucursal. Este formulario solo tiene un input: nombre de la sucursal. Una vez llenado este campo y accionado el botón para crear la sucursal, se realiza un llamado a la API, con un Request de tipo POST a la URL:

host/api/sucursales

En el body va el campo mencionado anteriormente. Al llegar la respuesta de la API, se muestra una notificación para mostrar si la sucursal fue creada con éxito o hubo un error (nombre de sucursal ya existente). Además, se actualiza el state en Vuex del array de sucursales a través de una mutación.

2. Tabla de sucursales: Este componente se encarga de listar las sucursales registradas. Este componente hace una llamada a la API a través de un Request de tipo GET a la URL:

host/api/sucursales

Al llegar el Response, se guarda el array de sucursales en el state del store de Vuex. Luego se recorre el arreglo para llenar el cuerpo de la tabla. Si el Response es un arreglo vacío, se muestra solo los encabezados y un mensaje confirmando

que no existen sucursales.

3. Formulario creación de niveles: Este componente es el encargado de crear un nuevo nivel pre-escolar. Este formulario solo tiene un input: nombre del nivel pre-escolar. Una vez llenado este campo y accionado el botón para crear el nivel se realiza un llamado a la API, con un Request de tipo POST a la URL:

host/api/niveles

En el body va el campo mencionado anteriormente. Al llegar la respuesta de la API se muestra una notificación para mostrar si el nivel fue creado con éxito o hubo un error (nombre de nivel ya existente). Además, se actualiza el state en el store de Vuex del array de niveles a través de una mutación.

4. Tabla de niveles: Este componente se encarga de listar los niveles registrados. Este componente hace una llamada a la API a través de un Request de tipo GET a la URL:

host/api/niveles

Al llegar el Response, se guarda el array de niveles en el state del store de Vuex. Luego se recorre el arreglo para llenar el cuerpo de la tabla. Si el Response es un arreglo vacío, se muestra solo los encabezados y un mensaje confirmando que no existen niveles.

Los dos componentes de tablas tienen la mismas acciones, editar y eliminar sucursal o nivel según corresponda.

Para editar una sucursal o nivel, se hace click sobre el registro que se quiere editar. Esto levanta un modal, en el cual aparece el nombre de la sucursal o nivel seleccionado. Al presionar el botón enviar, se realiza un Request de tipo PUT a la URL:

host/api/sucursales/:sucursalId

host/api/niveles/:nivelId

dependiendo de lo que se está editando. En el body se envía el nombre de la sucursal o del nivel, dependiendo del registro que se está editando. Al llegar el Response, a través de una mutación de Vuex se cambia el state del array de sucursales o niveles con el objeto de la sucursal o nivel que ha sufrido cambios. Finalmente, se muestra un mensaje de éxito o de error. Para eliminar una sucursal o nivel, se presiona el botón eliminar ubicado en la columna de acciones. Se realiza un llamado a la misma URL, pero esta vez la acción es de tipo DELETE.

La figuras 4.6, 4.7 y 4.8 muestran los diagramas flujo para el formulario de creación, tabla y acciones de variables.

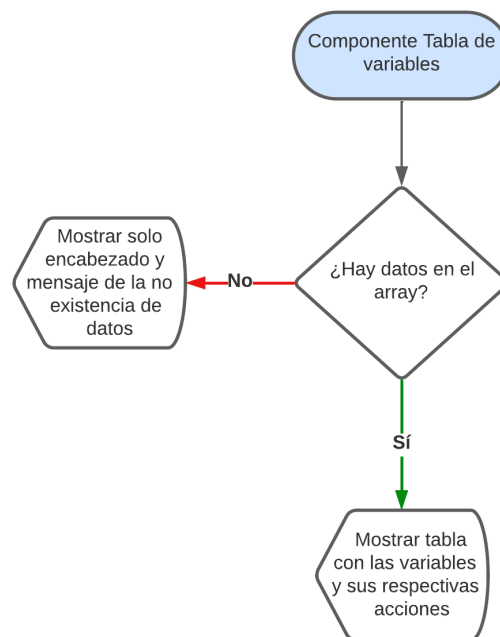


Figura 4.6: Diagrama de flujo del componente tabla variables

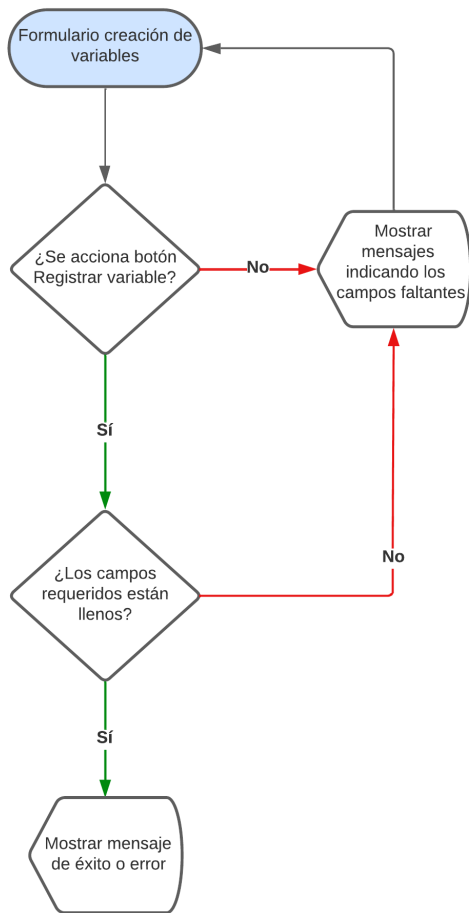


Figura 4.7: Diagrama de flujo formulario creación variables

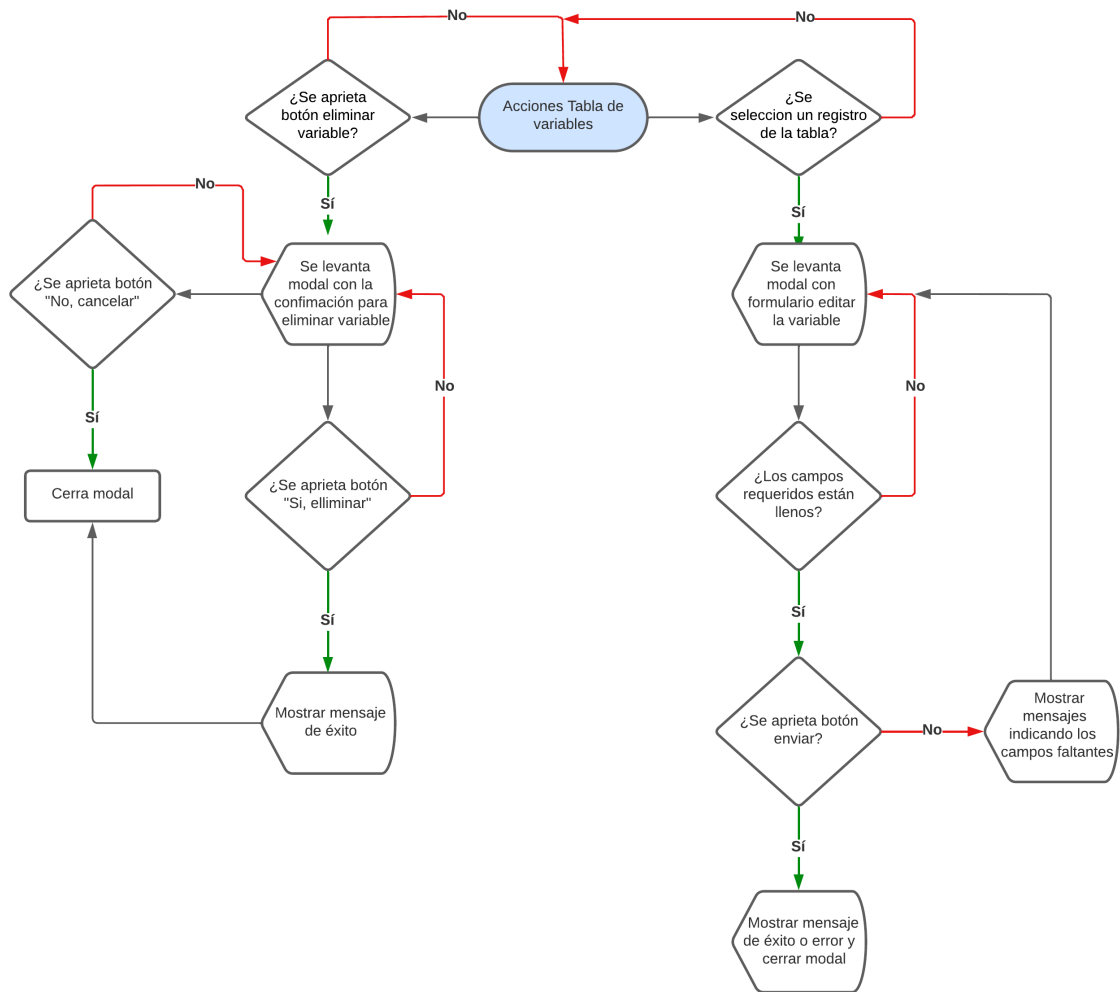


Figura 4.8: Diagrama de flujo de las acciones de tabla variables

Variables alimentarias

Al accionar el botón Variables Alimentarias, éste redirige hacia la vista donde se pueden crear y listar las variables de tipo de colación, platos e ingredientes.

Esta vista es similar a la vista analizada anteriormente, se cargan componentes de tablas con sus respectivos formularios de creación, solo cambian los URL de las APIs donde se realizan las acciones en el backend.

Copiar minutas

Cuando es accionado en el menú de navegación el botón Copiar minutas, éste redirige a la vista donde es posible copiar minutas de un nivel y tipo de colación en específico de una fecha seleccionada.

Esta vista solo carga un componente: formulario de copia de minutas. Este componente está compuesto por dos componentes QSelect ⁴, para seleccionar el nivel pre-escolar y tipo de colación a copiar. Además, se encuentran dos componentes QDate ⁵ (calendarios), en uno se selecciona el rango de fechas a copiar y en el otro se selecciona el primer día de rango de fechas a copiar. Cabe mencionar, que para copiar minutas deben de existir las matriculas en el mes a copiar.

Una vez ingresado los datos y es accionado el botón COPIAR MINUTAS, se realiza una llamada a la API de tipo POST a la URL:

host/api/minutas/copiar

El objeto del body tiene la siguiente estructura: fecha inicial del rango a copiar, fecha final del rango a copiar, fecha inicial del rango a pegar, ID nivel escolar, ID tipo colación e ID sucursal.

Cuando llega el Request, se muestra una notificación para mostrar si las minutas fueron copiadas con éxito o hubo algún problema (matrículas no creadas).

⁴<https://quasar.dev/vue-components/select>

⁵<https://quasar.dev/vue-components/date>

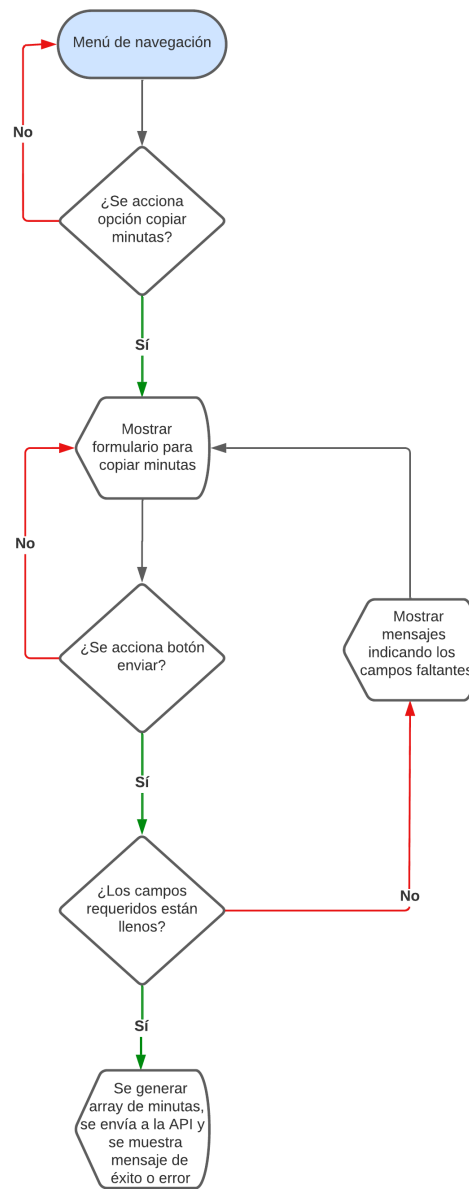


Figura 4.9: Diagrama de flujo de la vista copiar minutos

4.6. Resumen de vistas

Como se ha mencionado anteriormente, existen 3 distintos tipos de usuarios, por lo cual, cada usuario tiene acceso a algunas vistas de la plataforma dependiendo de su

tipo.

El cuadro 4.8, muestra a las funcionalidades que tiene acceso cada uno de los usuarios en la vista Home. Mientras que el cuadro 4.9 muestra a qué vistas de la plataforma tiene acceso cada uno de los 3 tipos de usuarios.

Funcionalidad	Administrador	Nutricionista	Manipulador de alimentos
Navegación en calendario	✓	✓	✓
Cambiar nivel	✓	✓	✓
Guardar minutas en PDF	✓		
Guardar calendario en PDF	✓		
Crear minuta	✓	✓	
Borrar minuta	✓	✓	
Ver minuta	✓	✓	✓
Editar n° de matrículas	✓		
Ver lista de compra	✓		
Cerrar Sesión	✓	✓	✓

Cuadro 4.8: Cuadro resumen de las acciones de la vista Home con respecto al rol de usuario

Vista	Administrador	Nutricionista	Manipulador de alimentos
Home	✓	✓	✓
Registra usuario	✓		
Listar usuarios	✓		
Crear variables institucionales	✓		
Crear variables alimenticias	✓	✓	
Stock	✓		
Matrículas	✓		
Copiar Minutas	✓		
Recetas	✓	✓	
Editar mis datos	✓	✓	✓

Cuadro 4.9: Cuadro resumen de acceso a las vistas de la plataforma

Capítulo 5

Resultados

En este capítulo, se muestra la implementación de la plataforma en un servidor web, las vistas de la plataforma y las disponibilidades que ha tenido la plataforma. Cabe mencionar, que se cuenta con la autorización de la Agencia Ingenium y del Jardín para adjuntar las imágenes de este capítulo. Además, las imágenes adjuntas de la plataforma son de un usuario con rol de administrador.

5.1. Implementación

Frontend

El Jardín y Sala Cuna Natgio, es cliente de la Agencia Ingenium; agencia de marketing digital, publicidad, diseño e implementación de página web a través de Wordpress. Natgio, cuenta con una página web que fue diseñada e implementada por esta agencia. La página web cuenta con el dominio natgio.cl y está alojada en los servidores de la agencia. A través de WHM (Web Host Manager), el cual es un panel de control que brinda al usuario la capacidad de administrar múltiples sitios basados en cPanel (panel de control basados en Linux para cuentas de hosting web), es posible administrar el servidor web de la agencia. Se crea un subdominio intranet.natgio.cl para alojar la plataforma. Además, cPanel permite generar de forma automática los certificados SSL,

el cual es un protocolo de seguridad que mantiene la información encriptada y segura en los sitios web que hagan uso de él, de esta forma, los usuarios saben que sus datos están protegidos al ingresarlos.

A través del comando de consola *quasar build*, es posible compilar el proyecto para producción. Si la compilación es exitosa, se genera una carpeta en el directorio raíz, llamada *dist*, que es la construcción de un archivo empaquetado, generalmente archivos estáticos como ****.js*, ****.css*, *index.html*, entre otros. Esta carpeta se carga en la ruta *public_html/intranet*, a través de cPanel.

La figura 5.1, muestra los subdominios y la ruta que tiene cada uno para acceder a los archivos en el servidor.

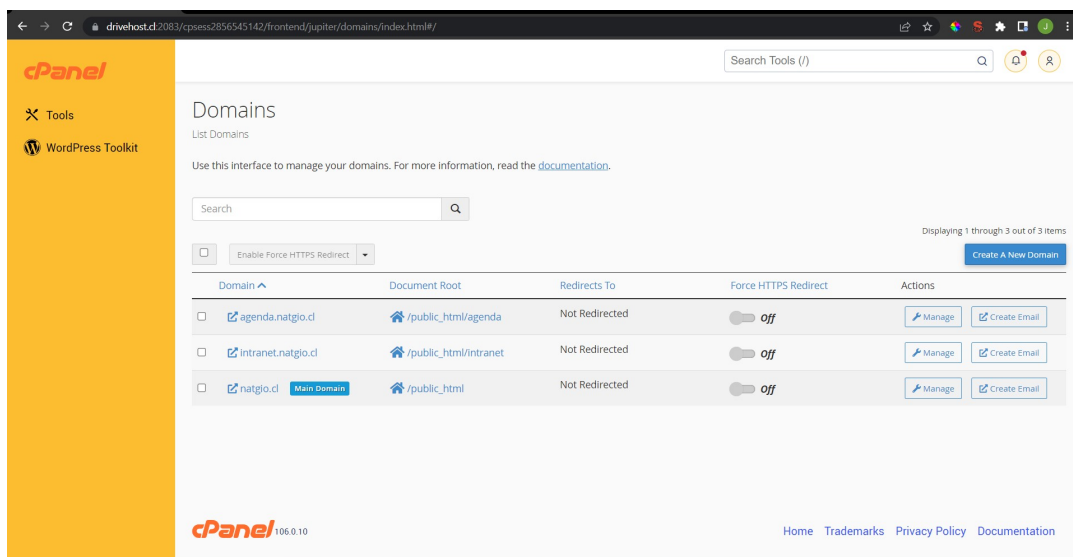


Figura 5.1: Subdominios de natgio.cl

Backend

Cpanel, permite crear base de datos MySQL a través phpMyAdmin. Se crea una base de datos, llamada *natgio_intranet*, en donde se alojarán todos los datos de la plataforma. La figura 5.2, muestra la base de datos de la plataforma.

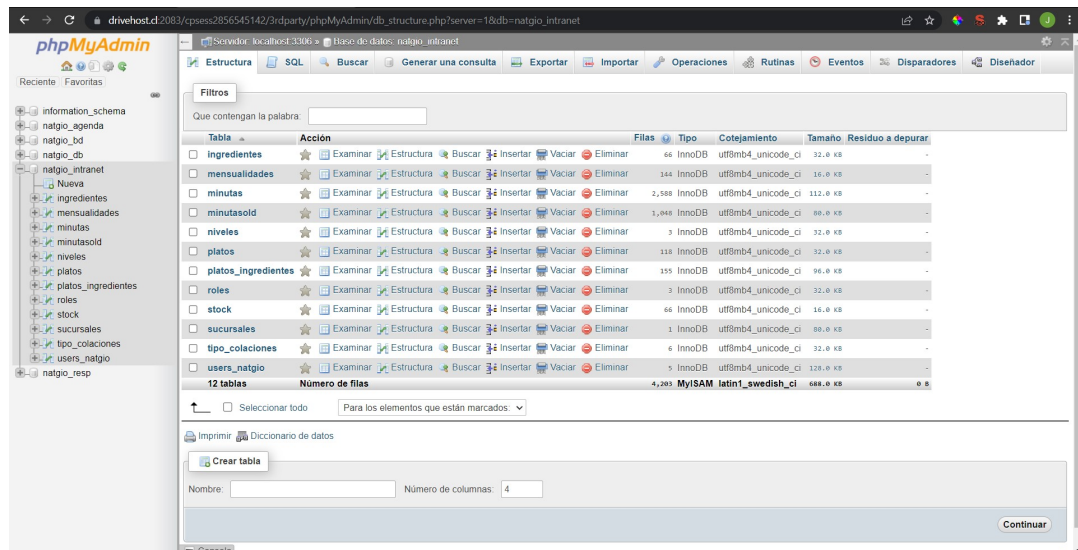


Figura 5.2: Base de datos de la plataforma a través de phpMyAdmin

La API de la plataforma, se compila en producción en Heroku, a través de comandos git por consola. Heroku ofrece una cuenta gratis para estudiantes, por lo cual, este servicio tendrá costo 0. La API, a través de Heroku, se conecta a la base de datos mencionada anteriormente. La figura 5.3, muestra el back-end alojado en Heroku.

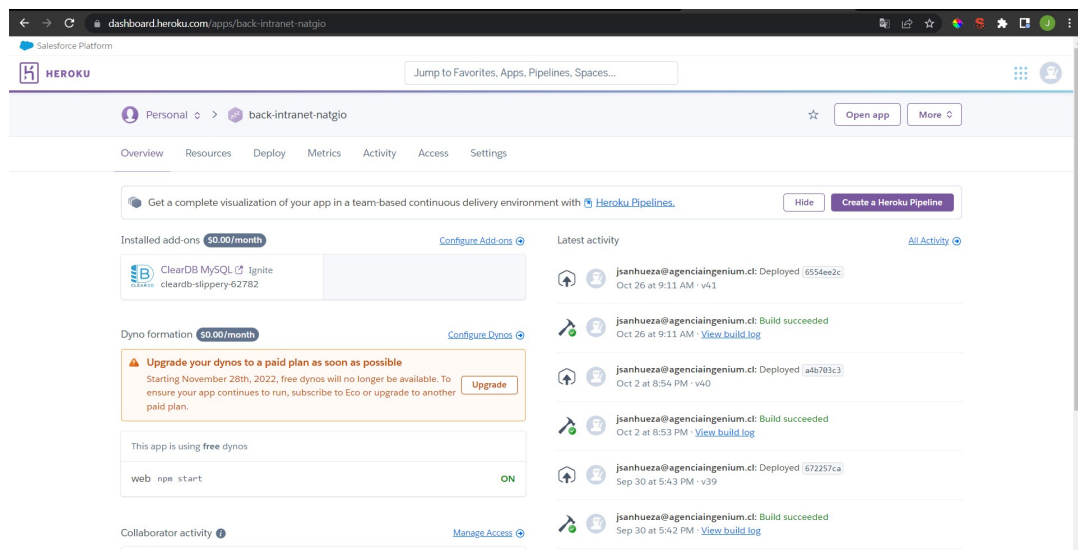


Figura 5.3: Backend de la plataforma alojado en Heroku

5.2. Disponibilidad de la plataforma

A través del servicio Freshping ¹, es posible medir la disponibilidad y tiempo promedio de respuesta de la plataforma. Además, es posible recibir notificaciones cuando la página o la API a sufrido caídas. Freshping comprueba si un sitio web está disponible haciendo ping a intervalos regulares, conocidos como intervalos de sondeo.

La disponibilidad se calcula:

$$\text{disponibilidad \%} = \frac{\text{tiempo de actividad total}}{\text{duración supervisada}}$$

donde:

Tiempo de actividad total % = Duración total monitoreada – tiempo de inactividad

La figura 5.4 muestra la disponibilidad de la página del 99,93 % y un tiempo promedio de respuesta del 258 ms.

La figura 5.5 muestra la disponibilidad de la API de un 100 % y un tiempo promedio de respuesta del 30 ms.

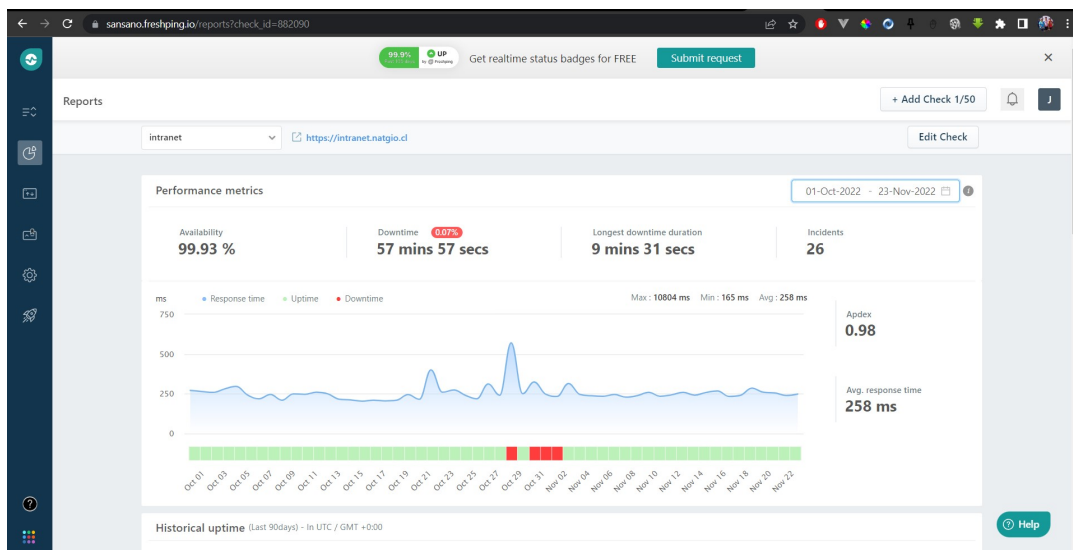


Figura 5.4: Reporte de la disponibilidad y tiempo de respuesta de la página donde está alojada la plataforma.

¹<https://www.freshworks.com/website-monitoring/>

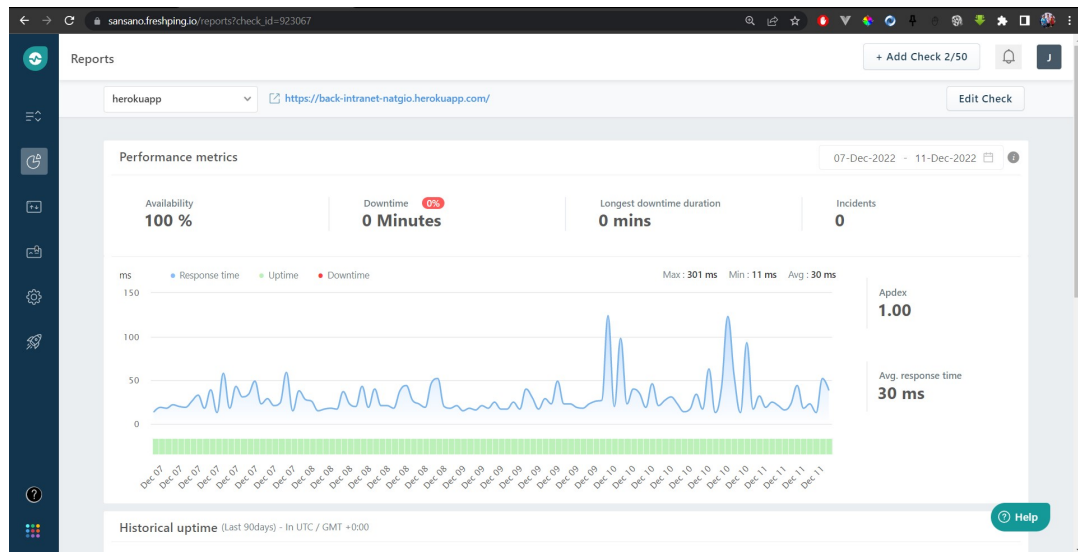


Figura 5.5: Reporte de la disponibilidad y tiempo de respuesta de la API en Heroku.

5.3. Imágenes de la aplicación web

En esta sección se presentan algunas capturas de pantalla de la aplicación ya en producción. Estas imágenes corresponden a la vista Home con algunas de sus funcionalidades. Para ver las otras vistas ver el anexo D.4.

5.3.1. Vista Home

Cuando se ha seleccionado la sucursal (Administrador), o se ha verificado que las credenciales ingresadas son correctas (Manipulador y Nutricionista), se redirecciona hacia la vista Home de la Figura 5.6, además, se muestra el menú de navegación de la plataforma. Como se mencionó en los cuadros 4.8 y 4.9 de la sección 4.6, las acciones en la vista Home y opciones del menú de navegación dependen del rol del usuario. Como se mencionó en la introducción de este capítulo, solo se muestran imágenes de la vista administrador, ya que tiene todas las acciones y vistas disponibles de la plataforma. En el menú de navegación es posible acceder a las demás vistas de la plataforma.

En la vista Home se encuentra el calendario, donde es posible crear (haciendo click

en el número del día del mes, ver figura 5.7) y visualizar las minutas (haciendo click sobre una receta del calendario, ver figura 5.8).

También es posible actualizar el número de matrículas desde los inputs en la esquina superior derecha de la figura 5.6. Además, es posible navegar por el calendario a través de las opciones *Hoy*, *<* y *>*. Estas opciones hacen que el calendario avance, retroceda un mes o vuelva a la fecha actual.

Al lado de las opciones mencionadas anteriormente, se encuentran las opciones de Guardar Minutas como PDF (figura 5.9), ver lista de compra (figura 5.10), cambiar a los demás niveles (los cuales muestran las mismas opciones solo cambia el nivel a mostrar) y guardar el calendario de minutas del nivel como PDF (figura 5.11).

Noviembre - 2022 Jardín Infantil					
Tipo Colación	Lunes	Martes	Miércoles	Jueves	Viernes
	31	1	2	3	4
Colación AM	Feriado	Feriado	Barra cereal	Flan	Jalea con yogurth
Ensalada	-	-	Tomate	Pepino	Betarraga
Principal	-	-	Porotos con riendas	Guiso zapallo italiano	Fideos
Acompañamiento	-	-	Sin Acompañamiento	Papas doradas	Salsa Alfredo
Postre	-	-	Fruta PM	Fruta PM	Fruta PM
Colación PM	-	-	Leche con Queque	Leche con Galleta Maria	Sin Colación PM
	7	8	9	10	11
Colación AM	Fruta AM	Babaraise	Flan	Jalea con yogurth	Yogurth con Fruta
Ensalada	Lechuga	Repollo	Pepino	Betarraga	Tomate
Principal	Lentejas guisadas	Carbonada Vacuno	Estofado de pollo	Garbanzos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Salsa Blanca con pollo
Postre	Fruta PM	Fruta PM	Fruta PM	Fruta PM	Fruta PM
Colación PM	Yogurth con Cereales	Leche con Queque	Leche con Galleta Integral	Leche con Galleta Integral	Sin Colación PM
	14	15	16	17	18
Colación AM	Fruta AM	Barra cereal	Flan	Fruta AM	Huevo duro
Ensalada	Betarraga	Repollo	Lechuga	Tomate	Pepino
Principal	Arvejas guisadas	Charquican de pavo	Tortilla de acelga/espinacas	Porotos con riendas	Fideos
Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Pure	Sin Acompañamiento	Salsa Bolognesa de vacuno
Postre	Fruta PM	Fruta PM	Fruta PM	Fruta PM	Fruta PM
Colación PM	Leche con Galleta Maria	Leche con Queque	Leche con Galleta Integral	Leche con Galleta Integral	Sin Colación PM
	21	22	23	24	25
Colación AM	Fruta AM	Mousse	Sémola con leche	Fruta AM	Jalea con yogurth
Ensalada	Repollo	Lechuga	Tomate	Betarraga	Pepino
Principal	Lentejas guisadas	Tomatitan con pollo	Guiso zapallo italiano	Garbanzos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Arroz Graneado	Arroz Graneado	Sin Acompañamiento	Salsa Blanca con pavo

Figura 5.6: Vista Home de la aplicación.

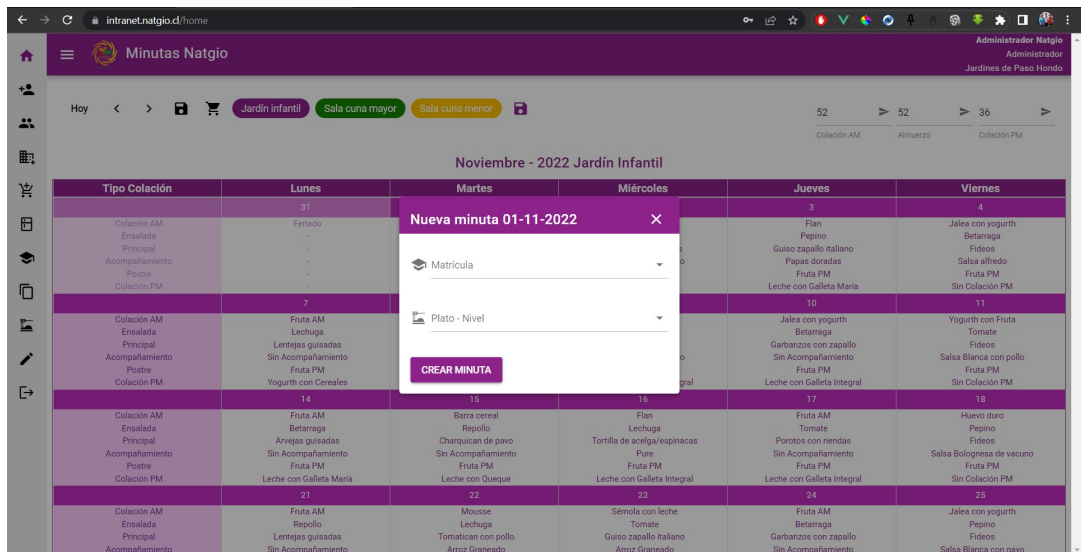


Figura 5.7: Modal creación de minutas.

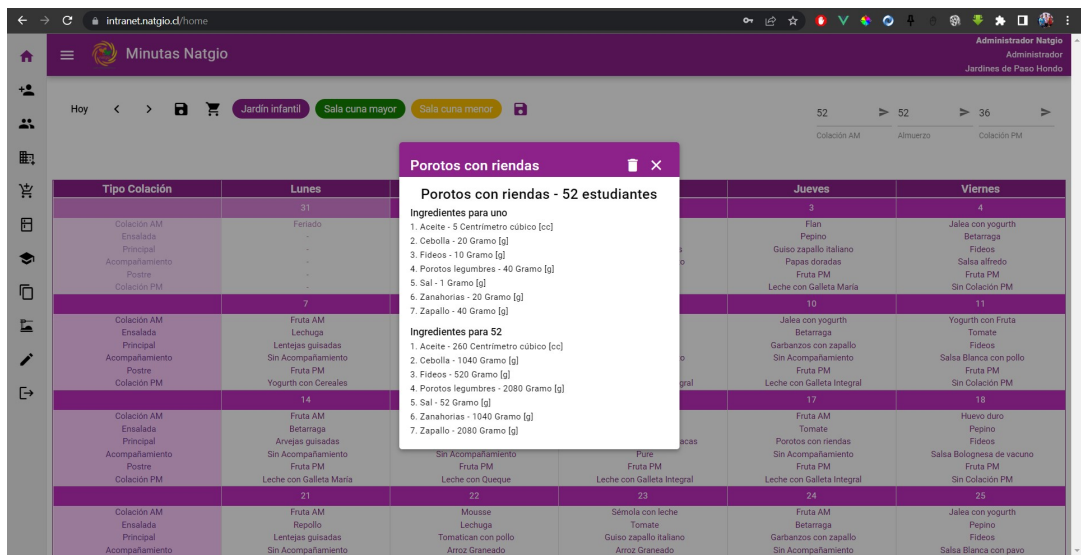


Figura 5.8: Modal visualización de minutas.

Fecha	Nivel	Plato	Ingredientes	Medidas por porción
2/11/2022	Jardin infantil	Porotos con riendas	Aceite Cebolla Fideos Porotos legumbres Sal Zanahorias Zapallo	5 Centrimetro cúbico [cc] 20 Gramo [g] 10 Gramo [g] 40 Gramo [g] 1 Gramo [g] 20 Gramo [g] 40 Gramo [g]
2/11/2022	Jardin infantil	Sin Acompañamiento		
2/11/2022	Jardin infantil	Fruta PM	Fruta	50 Gramo [g]
2/11/2022	Jardin infantil	Leche con Queque	Agua Leche semidescremada Queque	180 Centrimetro cúbico [cc] 28 Gramo [g] 1 Unidad
2/11/2022	Sala cuna mayor	Fruta AM	Fruta	80 Gramo [g]
2/11/2022	Sala cuna mayor	Fruta PM	Fruta	40 Gramo [g]
2/11/2022	Sala cuna mayor	Formula Láctea	Formula lactea personal	1 Unidad
2/11/2022	Sala cuna mayor	Porotos con riendas	Aceite Cebolla Fideos Porotos legumbres Sal Zanahorias Zapallo	5 Centrimetro cúbico [cc] 20 Gramo [g] 10 Gramo [g] 40 Gramo [g] 1 Gramo [g] 20 Gramo [g] 40 Gramo [g]
2/11/2022	Sala cuna mayor	Sin Acompañamiento		
2/11/2022	Sala cuna menor	Papilla Fruta estación	Fruta	80 Gramo [g]
2/11/2022	Sala cuna menor	Sin Ensalada		
2/11/2022	Sala cuna menor	Papilla Pollo con verduras	Aceite Acelga Fideos Papas Pollo	5 Centrimetro cúbico [cc] 25 Gramo [g] 10 Gramo [g] 50 Gramo [g] 15 Gramo [g]

Figura 5.9: Minutas en PDF.

Lista de comprar para Jardines de Paso Hondo - Noviembre 2022

Percible

Ingredientes	S1 02-11-2022 al 08-11-2022	S2 09-11-2022 al 15-11-2022	S3 16-11-2022 al 22-11-2022	S4 23-11-2022 al 29-11-2022	S5 30-11-2022 al 06-12-2022
Acelga	510 Gramo [g]	1.765 Gramo [g]	6.445 Gramo [g]	1.810 Gramo [g]	5.970 Gramo [g]
Albahaca	0	0	0	0	0
Betarraga	2.600 Gramo [g]	5.200 Gramo [g]	0	2.600 Gramo [g]	2.600 Gramo [g]
Brócoli	0	0	0	0	0
Cebolla	5.200 Gramo [g]	6.500 Gramo [g]	6.500 Gramo [g]	6.500 Gramo [g]	1.300 Gramo [g]
Fruta	28.400 Gramo [g]	28.400 Gramo [g]	32.040 Gramo [g]	31.520 Gramo [g]	4.640 Gramo [g]
Lechuga	2.600 Gramo [g]	0	5.200 Gramo [g]	2.600 Gramo [g]	0
Papas	9.040 Gramo [g]	6.970 Gramo [g]	990 Gramo [g]	6.310 Gramo [g]	270 Gramo [g]
Pepino	2.600 Gramo [g]	4.680 Gramo [g]	4.680 Gramo [g]	2.600 Gramo [g]	0
Repollo	2.080 Gramo [g]	2.080 Gramo [g]	2.080 Gramo [g]	2.080 Gramo [g]	0
Tomate	4.550 Gramo [g]	2.470 Gramo [g]	6.240 Gramo [g]	2.470 Gramo [g]	0
Zanahorias	5.870 Gramo [g]	8.415 Gramo [g]	7.245 Gramo [g]	9.250 Gramo [g]	1.300 Gramo [g]
Zapallo	8.360 Gramo [g]	9.630 Gramo [g]	5.860 Gramo [g]	9.790 Gramo [g]	150 Gramo [g]
Zapallo italiano	5.980 Gramo [g]	0	0	5.980 Gramo [g]	0

Supermercado

Ingrediente	Cantidad
Aceite	17.765 Centrimetro cúbico [cc]
Agua	178.880 Centrimetro cúbico [cc]
Arroz	13.120 Gramo [g]
Arvejas congeladas	3.380 Gramo [g]

Figura 5.10: Vista listado de compra.

Archivo | C:\Users\javie\Downloads\Minuta%20Jardin%20Infantil%20Noviembre%20-%202022.pdf

Minuta Jardín Infantil Noviembre - 2022.pdf 1 / 1 100%

Noviembre - 2022 Jardín Infantil					
Tipo Colación	Lunes	Martes	Miércoles	Jueves	Viernes
	31	1	2	3	4
Colación AM	Patado	Patado	Barra cereal	Flan	Jalea con yogurth
Ensalada	-	-	Tomate	Papino	Bermeja
Principal	-	-	Pomitos con verduras	Guiso zapallo italiano	Fideos
Acompañamiento	-	-	Sin Acompañamiento	Papas doradas	Salsa ahogada
Postre	-	-	Fruita PM	Fruita PM	Fruita PM
Colación PM	-	-	Leche con Quesque	Leche con Galleta María	Sin Colación PM
	5	6	7	8	9
Colación AM	Fruita AM	Batutanga	Flan	Jalea con yogurth	Yogurth con fruta
Ensalada	Lechuga	Repollo	Papino	Bermeja	Tomate
Principal	Letifapas guisadas	Cafionada Vacuno	Escalfado de pollo	Gabancos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Salsa Blanca con pollo
Postre	Fruita PM	Fruita PM	Fruita PM	Fruita PM	Fruita PM
Colación PM	Yogurth con Cereales	Leche con Quesque	Leche con Galleta Integral	Leche con Galleta Integral	Sin Colación PM
	11	12	13	14	15
Colación AM	Fruita AM	Barra cereal	Flan	Fruita AM	Huano duro
Ensalada	Bermeja	Repollo	Lechuga	Tomate	Papino
Principal	Arroz guisado	Charque de pavo	Tortilla de angu/huancas	Pomitos con verduras	Fideos
Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Papa	Sin Acompañamiento	Salsa Bolognesa de vacuno
Postre	Fruita PM	Fruita PM	Fruita PM	Fruita PM	Fruita PM
Colación PM	Leche con Galleta María	Leche con Quesque	Leche con Galleta Integral	Leche con Galleta Integral	Sin Colación PM
	16	17	18	19	20
Colación AM	Fruita AM	Mousse	Sémola con leche	Fruita AM	Jalea con yogurth
Ensalada	Bermeja	Lechuga	Tomate	Bermeja	Papino
Principal	Letifapas guisadas	Tomatitas con pollo	Guiso zapallo italiano	Gabancos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Arroz cremado	Arroz cremado	Sin Acompañamiento	Salsa Blanca con queso
Postre	Fruita PM	Fruita PM	Fruita PM	Fruita PM	Fruita PM
Colación PM	Leche con Galleta María	Yogurth con Cereales	Leche con Galleta Integral	Leche con Galleta María	Sin Colación PM
	21	22	23	24	25
Colación AM	Fruita AM	Mousse	Sémola con leche	Fruita AM	Jalea con yogurth
Ensalada	Bermeja	Lechuga	Tomate	Bermeja	Papino
Principal	Letifapas guisadas	Tomatitas con pollo	Guiso zapallo italiano	Gabancos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Arroz cremado	Arroz cremado	Sin Acompañamiento	Salsa Blanca con queso
Postre	Fruita PM	Fruita PM	Fruita PM	Fruita PM	Fruita PM
Colación PM	Leche con Galleta María	Yogurth con Cereales	Leche con Galleta Integral	Leche con Galleta María	Sin Colación PM
	26	27	28	1	2
Colación AM	Fruita AM	Huano duro	Arroz con leche	Fruita AM	Jalea con yogurth
Ensalada	Zanahoria	Lechuga	Bermeja	Bermeja	Papino
Principal	Arroz guisado	Charque de vacuno	Guiso angu/huancas	Gabancos con zapallo	Fideos
Acompañamiento	Sin Acompañamiento	Sin Acompañamiento	Papa	Sin Acompañamiento	Salsa Blanca con queso
Postre	Fruita PM	Fruita PM	Fruita PM	Fruita PM	Fruita PM
Colación PM	Yogurth con Cereales	Yogurth con Cereales	Leche con Quesque	Leche con Galleta Integral	Sin Colación PM

Figura 5.11: Calendario de minutas por nivel en PDF.

Capítulo 6

Conclusiones y Trabajos futuros

El desarrollo de la aplicación, para el sistema de minuta, se llevó a cabo utilizando el lenguaje de programación JavaScript; lenguaje elegido por tener un dominio previo de éste, gracias a las asignaturas de programación y prácticas de la carrera.

El frontend, backend y base de datos elegidos fueron:

- Quasar (basado en Vue.js): elegido por su conocimientos previo, su facilidad y rapidez para crear aplicaciones web.
- Express.js: elegido por su conocimiento previo y su fácil integración a servicios y middleware, lo cual, permitió aplicar de forma rápida la autenticación por token.
- Base de datos relacional: elegido debido a la estructura conocida de los datos.

Como la aplicación ha sido puesta en producción, el administrador ha podido utilizar la plataforma, de tal forma que lleva un tiempo creando minutas. Los problemas que han surgido han sido resueltos a la brevedad, para así cumplir los requerimientos de la aplicación.

La plataforma logró satisfacer las necesidades del administrador, permitiéndole optimizar los alimentos de la cocina; favoreciendo el manejo de éstos. Dicho en otras palabras, se ha optimizado el proceso de creación de minutas con una aplicación intuitiva, de fácil uso y una interfaz conocida para el usuario.

Esta plataforma se puede utilizar en cualquier institución que tenga un servicio de alimentación en sus dependencias, en donde los niveles educativos pueden ser cambiados por algún tipo de alimentación ya sea vegana, dietética, entre otros.

Con respecto a las perspectivas de masificación de la aplicación, queda a criterio del cliente, ya que él es el dueño de esta plataforma, pero es su idea presentar esta aplicación a otros jardines infantiles.

Para trabajos a futuro, queda por implementar un sistema de gestión interna del Jardín, donde se pueda ingresar información del personal, de los estudiantes, valores de matrículas, entre otros. La plataforma que se creó para las minutas alimenticias sería siendo un módulo del sistema de gestión a desarrollar.

Los requerimientos del Jardín Infantil y Sala Cuna Natgio, han sido desarrollados uno a uno, cumpliendo así los objetivos presentados en el capítulo 1.

Anexos

Anexos A

Definición y funcionamiento de los tokens

Uno de los métodos, para poder mantener asegurado el acceso a una API, es el de autenticación con tokens. Cuando un usuario se autentica en la plataforma, a través de su usuario y contraseña o a través de un proveedor (como puede ser Twitter, Facebook o Google); cada petición HTTP que haga el usuario irá acompañada de un Token en la cabecera. Este Token, es una cadena de caracteres que permite a la API identificar al usuario. El Token no se almacena en el servidor, si no, en el lado del cliente (por ejemplo en localStorage) y el API es el que se encarga de describir ese Token y de redirigir el flujo de la aplicación en un sentido u otro.

La autenticación por tokens de la plataforma para la creación de minutas, cuenta de los siguientes pasos:

1. Autenticación, usando usuario y contraseña.
2. Cuando es autenticado en el servidor, se genera una cadena de caracteres que contiene el token proporcionado por la biblioteca JWT ¹.
3. El token es enviado al cliente.

¹<https://jwt.io/libraries>

4. Almacenamiento del token en el lado del cliente (Vuex y localStorage).
5. Cada vez que se hace una petición, el token es enviado al servidor.
6. Se valida el token en el servidor y se otorga (o no) el acceso al recurso que el cliente a solicitado.

Los pasos mencionados anteriormente, se pueden ver en el diagrama de flujo de la figura A.1

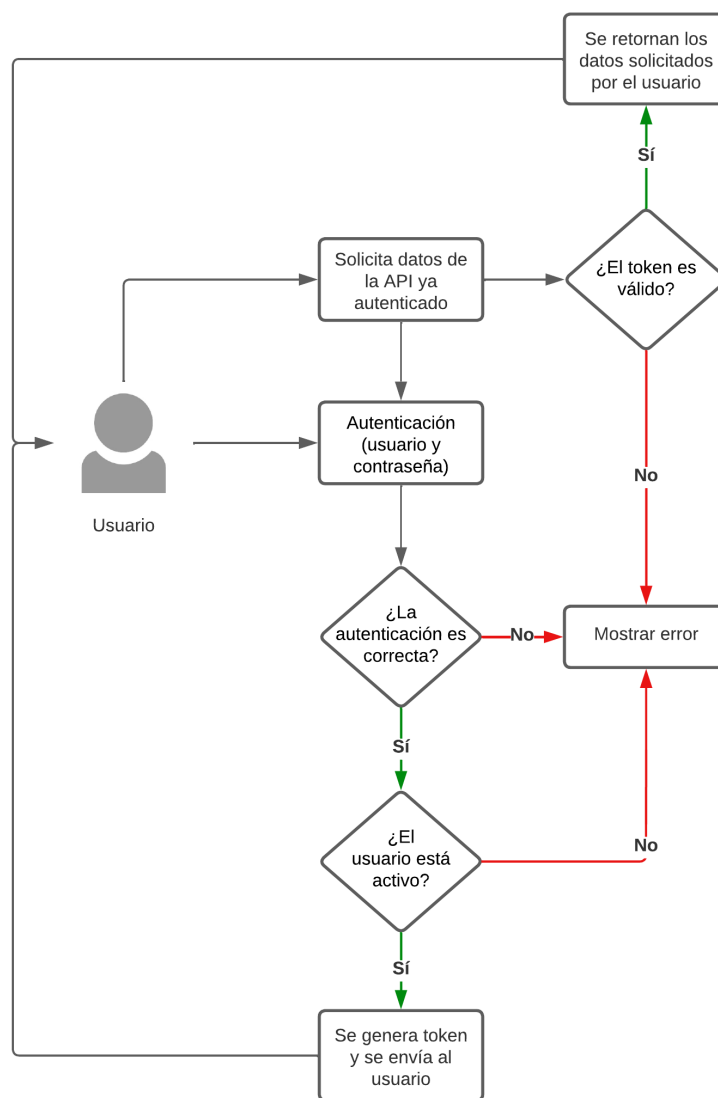


Figura A.1: Diagrama de flujo de los Token

Anexos B

Definición y funcionamiento de Vuex

La gestión de datos a nivel de aplicación, desempeña una de las funciones más importantes en la arquitectura y creación de aplicaciones web del lado del cliente. En Vue.js, existe la biblioteca Vuex ¹ para poder administrar los estados de la plataforma. Esta administración, puede verse como la capacidad de administrar datos compartidos entre componentes múltiples y a menudo no relacionados. Sirve como un almacén (store) centralizado para todos los componentes de una aplicación, con reglas que garantizan que el estado solo se puede modificar de forma predecible.

Existen 4 conceptos básicos para poder entender Vuex:

- **State:** El estado es un objeto plano, éste contiene los valores que definen la aplicación en un momento dado. Es la única fuente de verdad, es decir, es el único lugar donde consultar estos valores. Para poder hacer un cambio de estado, se debe hacer a través de las mutaciones.
- **Getters:** Es una función, en la cual se puede acceder al estado actual.
- **Mutations:** Como se mencionó con anterioridad, la única forma de alterar el estado es usando mutaciones. Cada mutación tiene un tipo (type) y un manejador (handler). El handler, es una función que recibe como primer argumento el estado

¹<https://vuex.vuejs.org/>

actual, y puede recibir opcionalmente un payload como segundo parámetro para poder modificar el estado.

- **Actions:** Las acciones, son las responsables de realizar las mutaciones, para poder modificar el estado. Además, es posible hacer operaciones asíncronas.

En la plataforma cada vez que se hace una llamada a la API con el método *fetch()*, se hace a través de una Actions. Cuando llega el Response con una respuesta positiva, los datos son almacenados en el State de la plataforma, a través de Mutations. Para poder acceder a estos datos en cualquier parte de la aplicación, se debe hacer a través de las funciones Getters.

En particular, el token del usuario es almacenado en Vuex, pero también es almacenado en el localStorage del navegador para mantener la sesión activa, aunque se cierre éste. Cabe destacar, que cuando el usuario cierra sesión, el token es eliminado del localStorage.

Anexos C

Endpoints de la aplicación

Ruta/ Descripción	Método	Header
<i>host/api/sucursales</i> Listado de sucursales	GET	Content-Type: application/json auth-token: token
<i>host/api/sucursales</i> Registrar una nueva sucursal	POST	Content-Type: application/json auth-token: token
<i>host/api/sucursales/:sucursal_id</i> Editar una sucursal	PUT	Content-Type: application/json auth-token: token
<i>host/api/sucursales/:sucursal_id</i> Traer la información de una sucursal	GET	Content-Type: application/json auth-token: token
<i>host/api/sucursales/:sucursal_id</i> Eliminar una sucursal	DELETE	Content-Type: application/json auth-token: token

Cuadro C.1: Tabla de endpoints de sucursales

Ruta/ Descripción	Método	Header
<i>host/api/users/login</i> Inicio de sesión	POST	Content-Type: application/json
<i>host/api/users/logout</i> Finalizar sesión	POST	Content-Type: application/json auth-token: token
<i>host/api/users</i> Registrar un nuevo usuario	POST	Content-Type: application/json auth-token: token
<i>host/api/users/:user_id</i> Editar usuario	PUT	Content-Type: application/json auth-token: token
<i>host/api/users/:user_id</i> Traer la información de un usuario	GET	Content-Type: application/json auth-token: token
<i>host/api/users/:user_id</i> Eliminar un usuario	DELETE	Content-Type: application/json auth-token: token
<i>host/api/users</i> Listado de los usuarios	GET	Content-Type: application/json auth-token: token
<i>host/api/users/sucursal/:sucursal_id</i> Listado de usuarios por sucursal	GET	Content-Type: application/json auth-token: token

Cuadro C.2: Tabla de endpoints de usuario

Ruta/ Descripción	Método	Header
<i>host/api/niveles</i> Listado de niveles	GET	Content-Type: application/json auth-token: token
<i>host/api/niveles</i> Registrar un nuevo nivel escolar	POST	Content-Type: application/json auth-token: token
<i>host/api/niveles/:nivel_id</i> Editar un nivel escolar	PUT	Content-Type: application/json auth-token: token
<i>host/api/sucursales/:nivel_id</i> Traer la información de un nivel escolar	GET	Content-Type: application/json auth-token: token
<i>host/api/niveles/:nivel_id</i> Eliminar un nivel escolar	DELETE	Content-Type: application/json auth-token: token

Cuadro C.3: Tabla de endpoints de niveles escolares

Ruta/ Descripción	Método	Header
<i>host/api/colaciones</i> Listado de tipo colación	GET	Content-Type: application/json auth-token: token
<i>host/api/colaciones</i> Registrar un nuevo tipo colación	POST	Content-Type: application/json auth-token: token
<i>host/api/colaciones/:colacion_id</i> Editar un tipo colación	PUT	Content-Type: application/json auth-token: token
<i>host/api/colaciones/:colacion_id</i> Traer la información de un tipo colación	GET	Content-Type: application/json auth-token: token
<i>host/api/colaciones/:colacion_id</i> Eliminar un tipo colación	DELETE	Content-Type: application/json auth-token: token

Cuadro C.4: Tabla de endpoints de tipo colación

Ruta/ Descripción	Método	Header
<i>host/api/platos</i> Listado de platos	GET	Content-Type: application/json auth-token: token
<i>host/api/platos</i> Registrar un nuevo plato	POST	Content-Type: application/json auth-token: token
<i>host/api/platos/:plato_id</i> Editar un plato	PUT	Content-Type: application/json auth-token: token
<i>host/api/platos/:plato_id</i> Traer la información de un plato	GET	Content-Type: application/json auth-token: token
<i>host/api/platos/:plato_id</i> Eliminar un plato	DELETE	Content-Type: application/json auth-token: token

Cuadro C.5: Tabla de endpoints de platos

Ruta/ Descripción	Método	Header
<i>host/api/recetas</i> Listado de recetas	GET	Content-Type: application/json auth-token: token
<i>host/api/recetas</i> Registrar una nueva receta	POST	Content-Type: application/json auth-token: token
<i>host/api/recetas/:receta_id</i> Editar una receta	PUT	Content-Type: application/json auth-token: token
<i>host/api/recetas/:receta_id</i> Traer la información de una receta	GET	Content-Type: application/json auth-token: token
<i>host/api/recetas/:recetas_id</i> Eliminar una receta	DELETE	Content-Type: application/json auth-token: token

Cuadro C.6: Tabla de endpoints de recetas

Ruta/ Descripción	Método	Header
<i>host/api/matriculas</i> Listado de matrículas	GET	Content-Type: application/json auth-token: token
<i>host/api/matriculas</i> Registrar una nueva matrícula	POST	Content-Type: application/json auth-token: token
<i>host/api/matriculas/:matricula_id</i> Editar una matrícula	PUT	Content-Type: application/json auth-token: token
<i>host/api/matriculas/:matricula_id</i> Traer la información de una matrícula	GET	Content-Type: application/json auth-token: token
<i>host/api/matriculas/:matriculas_id</i> Eliminar una matrícula	DELETE	Content-Type: application/json auth-token: token

Cuadro C.7: Tabla de endpoints de matrículas

Anexos D

Detalles y diagramas de flujo

D.1. Vista Selección de sucursal

Esta vista, es solo accesible para un usuario administrador, ya que los demás usuarios tienen su sucursal guardada en la base de datos. Para poder obtener la lista de sucursales, se realiza un Request de tipo GET a la URL *host/api/sucursales*.

Este listado serán las opciones del formulario de selección de sucursales. Una vez seleccionada una sucursal, ésta es guardada en el state, a través de Vuex, para luego ser redireccionado a la vista home.

La siguiente figura, muestra el diagrama de flujo de esta vista.

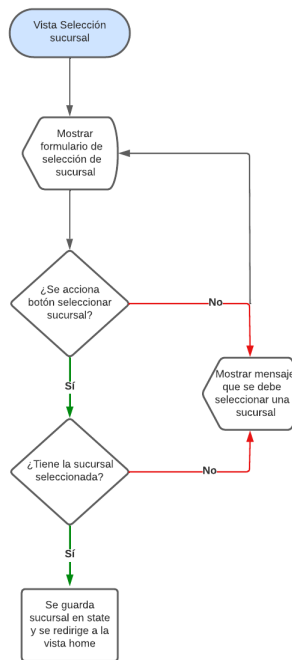


Figura D.1: Diagrama de flujo de la vista selección sucursal

D.2. Vista Home

Calendario

Este componente, carga las minutas de un nivel pre-escolar en un mes en específico. Ocupa el componente `QCalendarMonth` de la biblioteca `QCalendar`¹ de Quasar. Cuando se ingresa a la vista Home, se hace un llamado de tipo GET a la URL

host/api/minutas?filtros=sucursalId|inicioMes|finMes|nivelId

Se agrega un params de filtros, para que sólo traiga las minutas de la sucursal en sesión, de un nivel en específico, de un mes en particular. En este primer llamado a la API, se traen las minutas del nivel Jardín Infantil del mes actual.

Al llegar el Response de la API se verifica si el array de objetos de minutas no está

¹<https://quasarframework.github.io/quasar-ui-qcalendar/docs>

vacío. Al no estar vacío se procede a guardar este array en el state de minutos de Vuex, a través de una mutación. En caso contrario, se muestra una notificación a través de Notify para indicar que no existen minutos en el mes.

La siguiente figura muestra el flujo de funcionamiento básico del componente calendario.

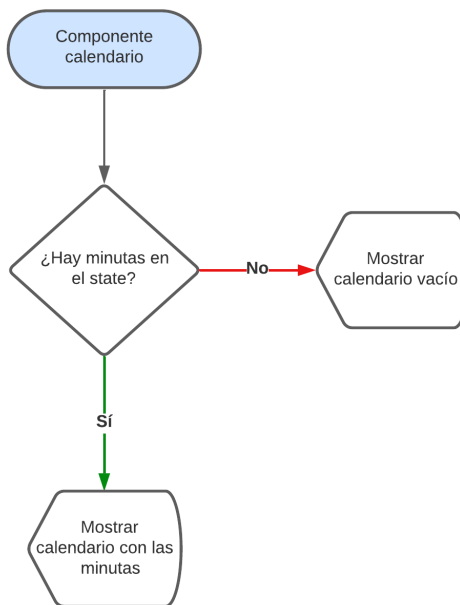


Figura D.2: Diagrama de flujo del componente de calendario

Este componente tiene las siguientes acciones:

- Navegación de calendario: Existen 3 botones para navegar en el calendario, *Hoy*, *<* y *>*. Cada una de estas opciones realiza el mismo llamado que se explicó anteriormente, cambiando el valor *inicioMes* y *finMes*. Cada vez que se realiza la llamada, se reemplaza en el state de minutos el array devuelto por la llamada a la API.
- Agregar minuta: Para agregar una minuta, es necesario hacer click en el día del mes donde se desea agregar la minuta. Esto levanta un modal con un formulario

de creación de minuta. Este formulario está compuesto por dos select: uno para ingresar la matrícula correspondiente al mes, nivel y tipo de colación, y el otro para ingresar la receta que tendrá la minuta. Al accionar el botón CREAR MINUTA (con los datos llenados), se realiza una llamada a la API de tipo POST a la URL:

host/api/minutas

El objeto del body de esta llamada tiene la siguiente estructura: fecha de la minuta, id de la receta, id de la matrícula e id de la sucursal.

Al llegar el Response desde la API, se muestra una notificación para indicar que la minuta ha sido creada. En este caso, el modal no se cierra, para permitir al usuario crear más minutas. Además, se agrega este objeto al array de minutas en el state de Vuex a través de una mutación.

La figura D.3 muestra el diagrama de flujo para la creación de minutas.

- Detalles de minuta: Para acceder a esta acción, es necesario hacer click en cualquier minuta del calendario. Esto levanta un modal, el cual hace un Request a la API de tipo GET a la URL:

host/api/minutas/:minuta_id

Este modal muestra la cantidad de ingredientes para solo un estudiante y para el número de matrículas correspondiente. Además, este modal tiene la opción de eliminar minuta, el cual hace un llamado a la API de tipo DELETE a la URL:

host/api/minutas/:minuta_id

Al llegar el Response, se hace una mutación al state de minutas en Vuex quitando el objeto de la minuta eliminada.

La figura D.4 muestra el diagrama de flujo para ver los detalles y eliminar una

minuta.

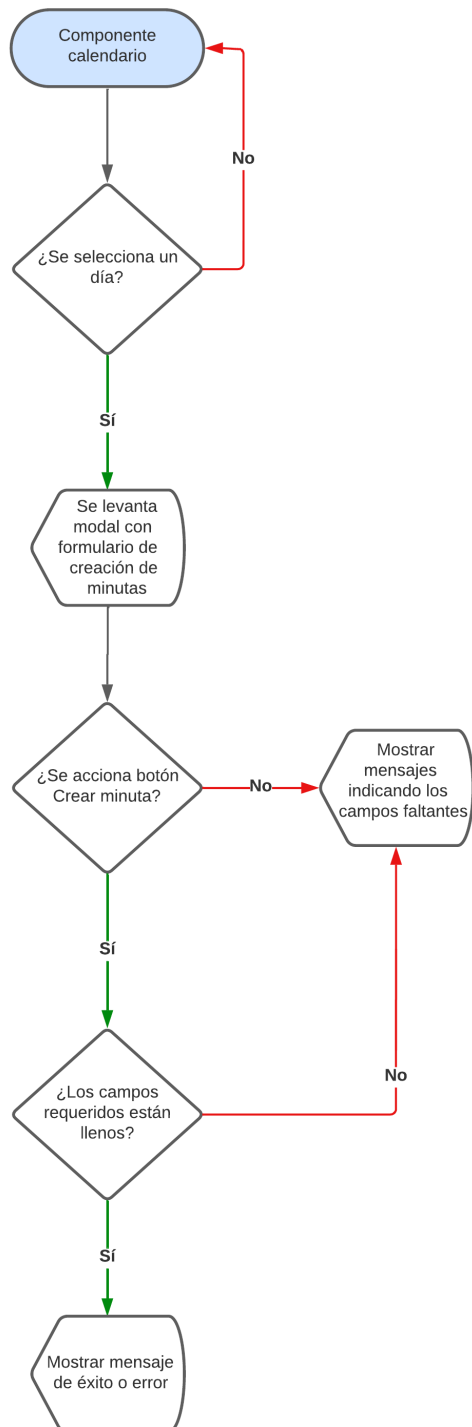


Figura D.3: Diagrama de flujo para crear minutos

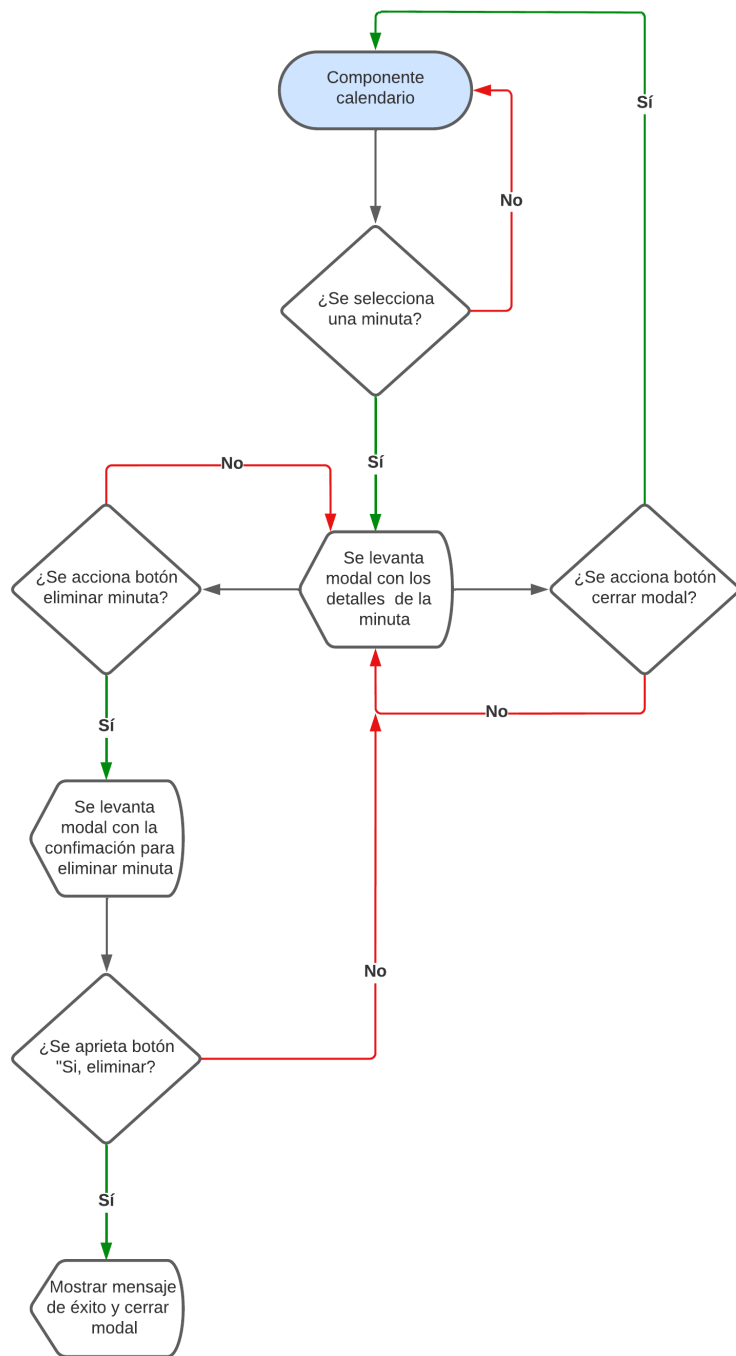


Figura D.4: Diagrama de flujo detalles minutas

D.2.1. Cambiar de nivel

Es posible cambiar de nivel pre-escolar, para poder hacer acciones al calendario correspondiente a ese nivel. Esto es posible a través de botones con los nombres de los niveles. Al hacer click en uno de estos botones, éste hace un llamado a la misma API cuando se ingresa a la vista Home, solo cambia el valor del params filtro *nivelId*.

La siguiente figura muestra el diagrama flujo de esta acción.

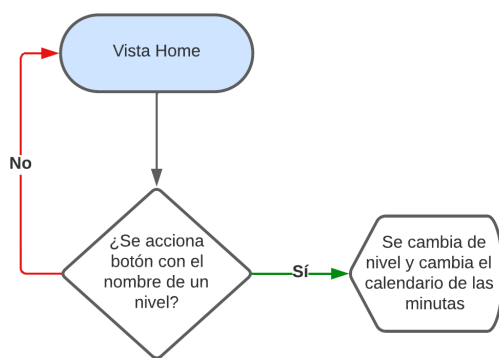


Figura D.5: Diagrama de flujo cambiar nivel

D.2.2. Lista de compra

Cuando es accionado el botón Lista de compra, éste redirige hacia la vista, donde es posible observar los ingredientes (Percible o Supermercado) que se deben comprar durante la semana o mes. Para aquello, se realiza una llamada a la API de tipo GET a la URL:

host/api/stock/lista-de-compra?filtros=sucursalID|mes|año.

Esta API, suma la cantidad usada de cada ingrediente en las minutos por el número de matrículas y le resta el stock actual del ingrediente. Cuando llega el Response, se carga el array de lista de compras en las dos tablas (Percible y Supermercado). En el

caso de la tabla Perecible, se muestran columnas con las semanas que tiene el mes y la cantidad del ingrediente a comprar. Mientras que en la tabla supermercado, se muestra una columna con el nombre del ingrediente y la cantidad a comprar. Si el array es vacío, se muestra la tabla con sus columnas y cantidad a comprar igual a 0.

La siguiente figura muestra el diagrama flujo de esta acción.

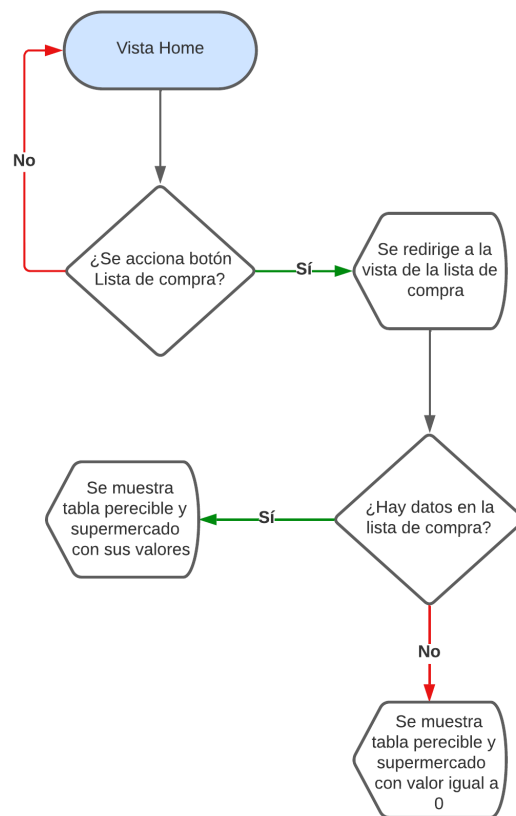


Figura D.6: Diagrama de flujo lista de compra

D.2.3. Cantidad de matrículas

Es posible editar el número de matrículas de una sucursal a través del de un formulario que consiste en 3 inputs, en los cuales se puede actualizar el número de matrículas

por tipo de colación y nivel. Cabe mencionar, que son 3 inputs debido a que uno es para la Colación AM, otro para el Almuerzo (Ensalada, Principal, Acompañamiento, Postre) y finalmente para la Colación PM. Cuando se carga un calendario, se cambia de nivel o mes, se realiza un Request de tipo GET a la URL:

host/api/matriculas?filtros=mes|anio|nivelId

Cuando llega el Response, se cargan esos valores en los inputs mencionados anteriormente. Si la respuesta es vacío, no se cargan valores. Para modificar uno de estos valores, se realiza un llamado del tipo PUT a la API en la URL:

host/api/matriculas/:matricula_id

.El body de esta llamada tiene la siguiente estructura: mes de la matrícula, año de la matrícula, cantidad de estudiantes en la matrícula, ID del Nivel escolar de la matrícula, ID de la sucursal en sesión e ID del tipo de colación.

Con la llegada del Response, se procede a mostrar una notificación con el éxito o error en la actualización de la matrícula.

La siguiente figura muestra el diagrama flujo de esta acción.

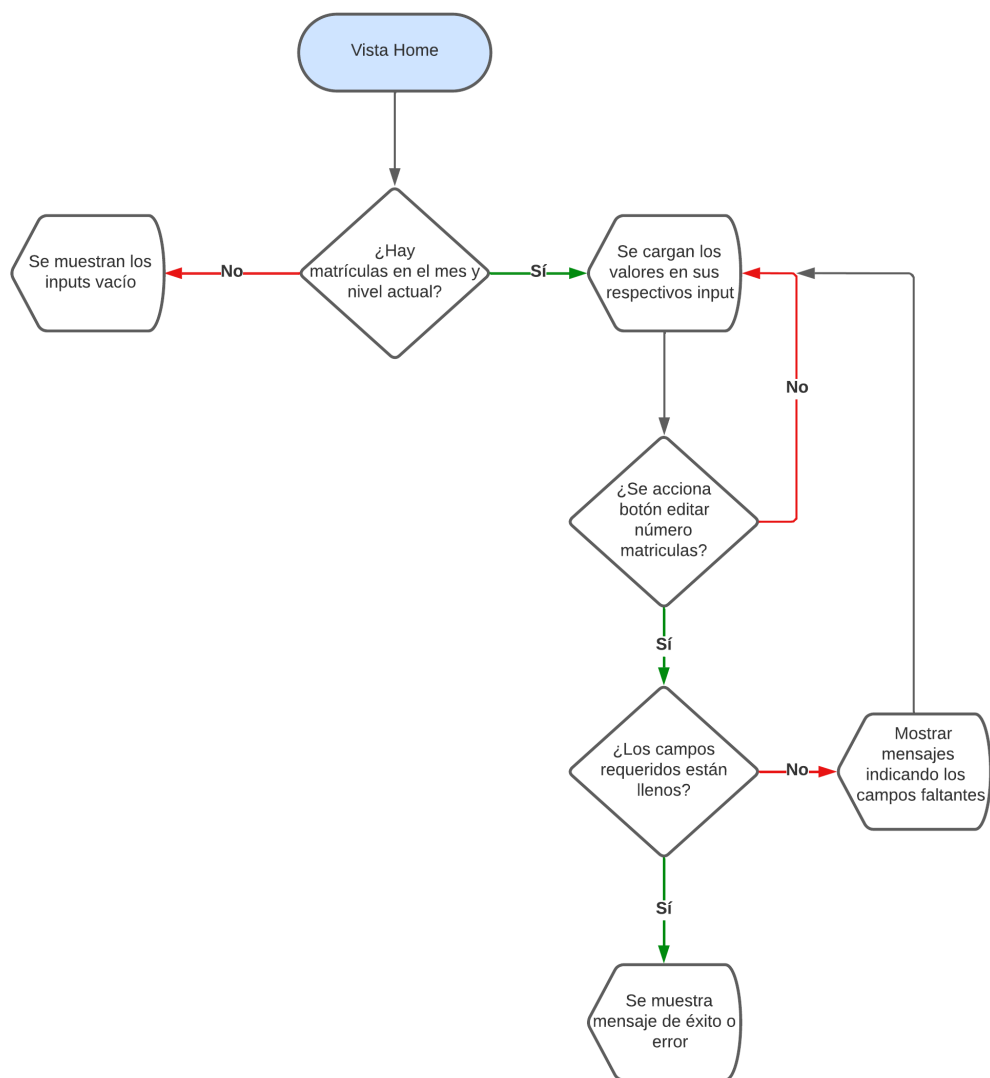


Figura D.7: Diagrama de flujo actualizar matrículas

D.3. Menú de navegación

D.3.1. Registrar usuario

Al accionar el botón Registrar, se redirige hacia la vista para registrar un nuevo usuario. Esta vista carga el un formulario, el cual contiene los siguientes inputs que el administrador debe llenar:

- Nombres
- Apellido.
- Correo Electrónico.
- Número de teléfono.
- Contraseña.
- Rol

Cuando se llena todo el formulario y se acciona el botón REGISTRAR, se hace un llamado a la API, con un Request de tipo POST a la URL:

host/api/users

En el body se mandan los datos rellenos del formulario, además de la ID de la sucursal donde se está registrando.

Al llegar la respuesta de la API se muestra una notificación con Notify de Quasar para mostrar si el usuario fue añadido con éxito o hubo algún error (email ya registrado). Este proceso se puede observar en el diagrama de flujo de la figura D.8

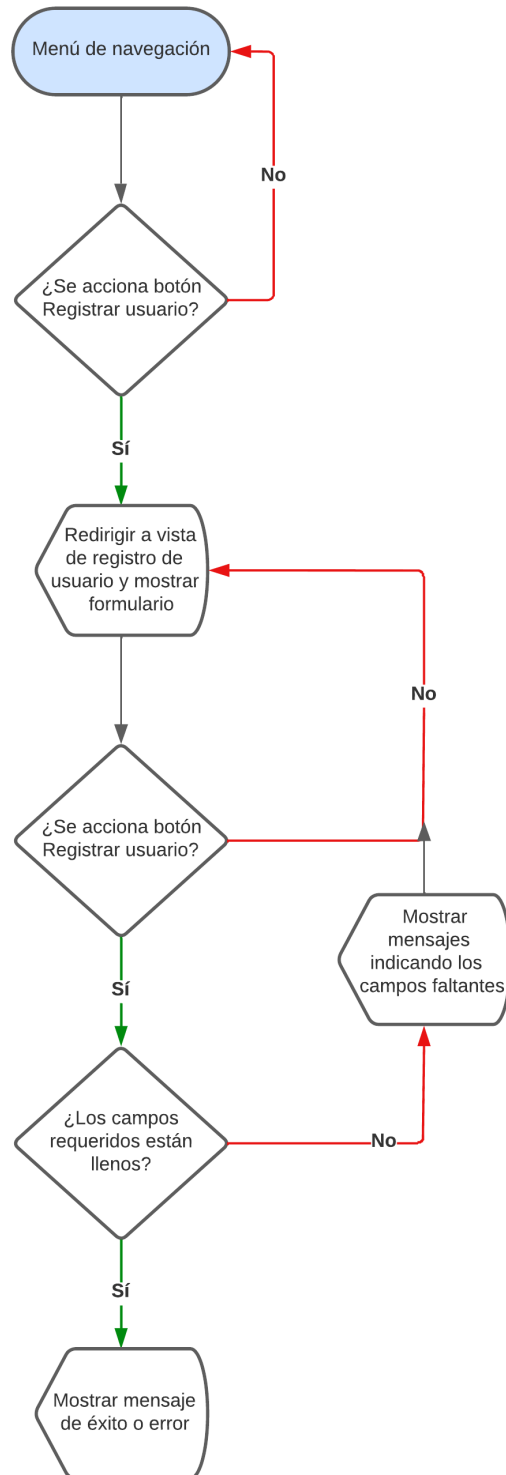


Figura D.8: Diagrama de flujo registro de un usuario.

D.3.2. Listar usuarios

Al accionar el botón Listar usuarios, éste redirige hacia la vista donde se listan todos los usuarios que pertenecen a la sucursal en sesión. Esta vista carga un componente QTable² de Quasar. Este componente hace una llamada a la API a través de un Request del tipo GET a la URL:

host/api/users?filtro=sucursalId

En esta ocasión, se envía un params filtro con el id de la sucursal que está en sesión a través de la URL, para poder traer solo los usuarios de la sucursal.

Al llegar el Response de la API, se guarda este array de usuarios en el store de Vuex. Luego se recorre el arreglo para poder rellenar el cuerpo de la tabla con la información de los usuarios de la sucursal. Si la respuesta de la API es un arreglo vacío, se muestra la tabla con sus encabezados y un mensaje confirmando que no existen usuarios en la sucursal.

Dentro de la tabla, existe una columna llamada acciones con las siguiente opciones:

- Activar/Desactivar usuario: En esta acción se hace un Request de tipo PUT a la URL:

host/api/users/:userId

indicando en la URL el id del usuario. En el body se manda todos los datos de usuario, solo cambia el status del usuario. Al llegar el Response, a través de una mutación de Vuex se cambia el state del array de usuarios con el objeto del usuario que ha sufrido cambios. Finalmente, se muestra una notificación con la acción que se ha hecho anteriormente.

- Cambiar contraseña: El administrador es capaz de cambiar la contraseña a cualquier usuario de la sucursal. Al apretar esta opción, se levanta un componente

²<https://quasar.dev/vue-components/table>

QDialog ³, el cual es un modal, con dos inputs (Contraseña y Confirmar contraseña). Se ocupa el mismo tipo de Request y URL, solo que en esta acción en el body se agrega la password a modificar. Al llegar el Response de la API, se muestra un mensaje de éxito o de error.

- Eliminar usuario: En esta acción se hace un Request de tipo DELETE a la URL:

host/api/users/:userId

indicando en la URL el id del usuario. Al apretar este botón se levanta un componente QDialog con la confirmación de esta acción. Al llegar el Response de la API, se realiza una mutación con Vuex para poder eliminar del state del array de usuarios el objeto eliminado. Finalmente, se muestra un mensaje de éxito o de error.

³<https://quasar.dev/vue-components/dialog>

En la figura D.9 y D.10 se puede ver el diagrama de flujo del componente y el diagrama de flujo de las acciones de la tabla.

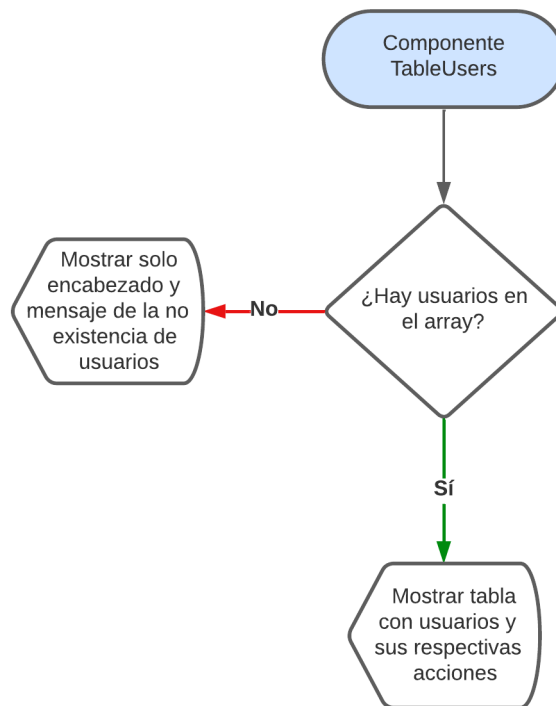


Figura D.9: Diagrama de flujo del componente tabla de usuarios

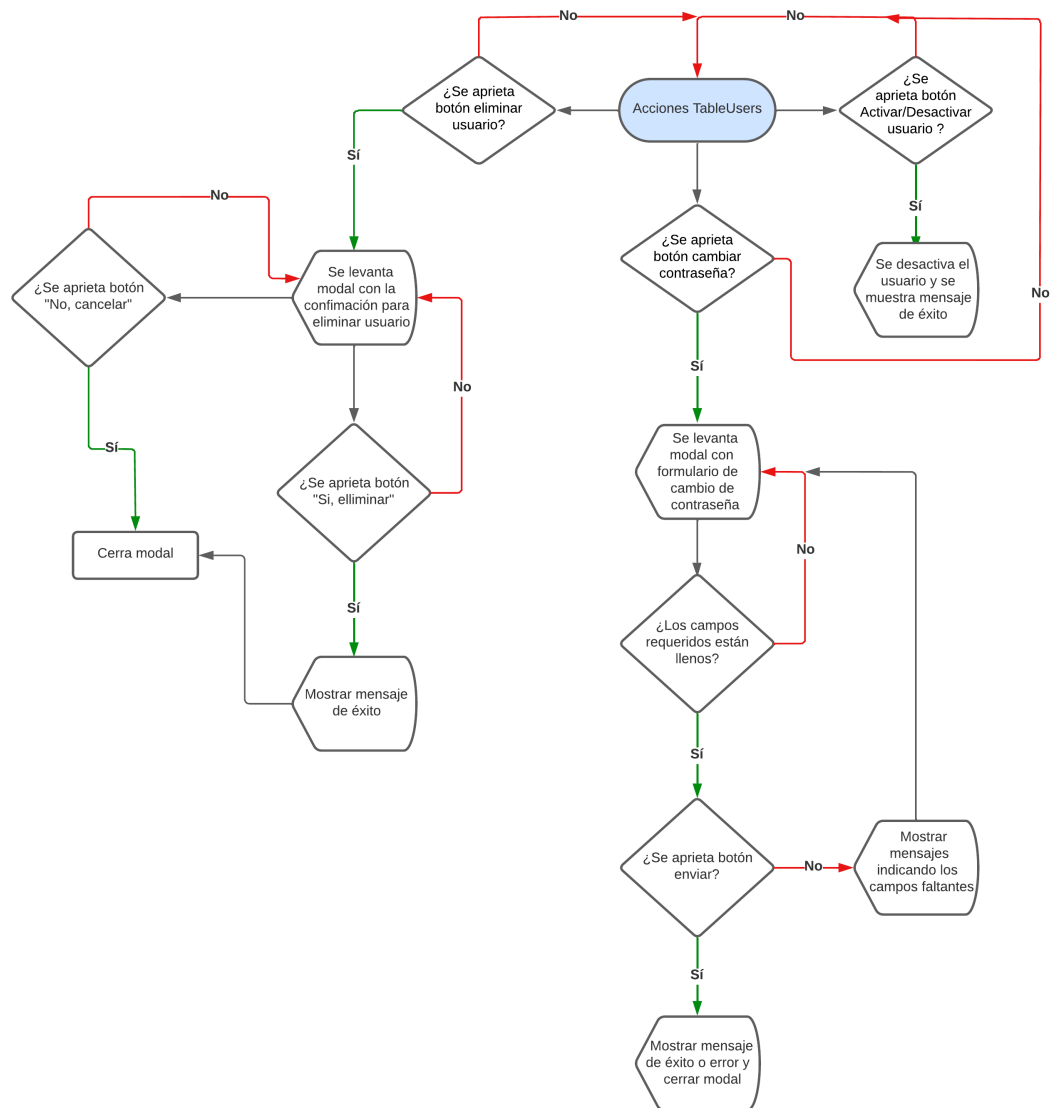


Figura D.10: Diagrama de flujo del acciones del componente tabla de usuarios

D.3.3. Stock

Cuando en el menú de navegación es accionado el botón Stock, éste redirige hacia la vista donde se puede observar y editar el Stock de la sucursal en sesión.

Como son dos tipos de ingredientes, en esta vista se presentan dos tablas con stock.

Al ingresar a la vista, se hace un llamado a la API a través de un Request del tipo

GET a la URL:

host/api/stock?filtro=sucursalId

Se hace envío del params filtro con el id de la sucursal, para que en la respuesta de la API solo venga el stock de la sucursal en sesión. Cuando llega el Response, el array de objetos de stock se guarda en el state de stock, en el store de Vuex. Luego, se recorre el arreglo para poder llenar el cuerpo de la tabla con la información del stock del ingrediente de cada tabla (Ingrediente Supermercado e Ingrediente Perecible). Si la respuesta de la API es un arreglo vacío, se muestra la tabla con sus respectivos encabezados (ingrediente, stock y unidad de medida).

Las acciones de estas tablas permiten editar el stock del ingrediente y la unidad de medida que tiene. Para editar el stock, se hace click sobre la cantidad de stock del ingrediente, por otra parte, para editar la unidad de medida se hace click sobre la unidad de medida del registro. Estas acciones levantan un modal, en el cual se puede editar el valor actual del stock y seleccionar la unidad de medida, las cuales pueden ser: Unidad, Gotas, Centímetros cúbicos y Gramos. Cuando se presiona el botón enviar, se realiza un Request de tipo PUT a la URL:

host/api/stock/:stockId

En el body se envía el objeto del stock. Al llegar el Response, a través de una mutación de Vuex, se cambia el state del array con el objeto editado. Luego, se muestra un mensaje de éxito o error.

La figura 4.7 de la vista de creación de variables, es un ejemplo de diagrama de flujo para el componente de TablaStock, mientras que la figura D.11 muestra el diagrama de flujo para las acciones de esta tabla.

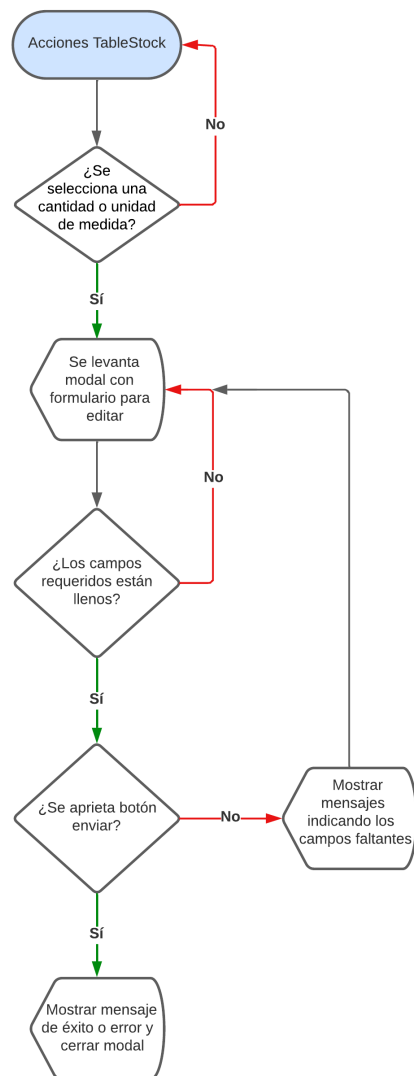


Figura D.11: Diagrama de flujo de las acciones de tabla stock

D.3.4. Matrículas

Al apretar desde el menú de navegación el botón Matrículas, éste redirige a la vista donde es posible crear, observar y editar las matrículas de la sucursal en sesión.

En esta vista se cargan dos componentes que son analizados a continuación.

1. Formulario de creación de matrículas: En este formulario se encuentran dos com-

ponentes QSelect ⁴ los cuales permiten seleccionar el mes y año de las matrículas a crear. Al accionar el botón GENERAR MATRÍCULAS (con los campos mes y año ya seleccionados), se crea un array con las matrículas a enviar a la API. Este array es un arreglo de objeto matrículas, que se genera a través de los niveles pre-escolares y tipo de colaciones. Por lo cual, si existen 3 niveles pre-escolares y 6 tipos de colaciones, serán 18 objetos dentro del array. Cada objeto tiene la siguiente estructura: ID sucursal, mes, año, ID nivel pre-escolar e ID tipo colación. Luego, se realiza la llamada a la API a la URL:

host/api/matriculas

de tipo POST. En el body se adjunta el array, creado anteriormente. Cuando llega el Request se muestra una notificación para mostrar si las matrículas fueron guardadas con éxito o si hubo algún error (matrículas ya existentes en la base de datos). Si las matrículas fueron guardadas con éxito, se actualiza en Vuex el state del array de matrículas a través de una mutación.

2. Tabla de matrículas: Este componente se encarga de listar las matrículas desde el mes en curso hacia adelante de la sucursal en sesión. Para esto, se realiza una llamada a la API a través de un Request de tipo GET a la URL:

host/api/matriculas

Cuando llega el Response, el array de matrículas es guardado en el State del store de Vuex. Se recorre el arreglo para llenar el cuerpo de la tabla. Si el Response es un arreglo vacío, se muestra solo los encabezados y un mensaje confirmando que no existen ingredientes.

Las acciones de esta tabla permiten al usuario editar la cantidad de estudiantes por matrícula, eliminar la matrícula (las cuales son similares a las analizadas en el variables

⁴<https://quasar.dev/vue-components/select>

institucionales) y filtrar por niveles pre-escolares. En esta tabla es posible filtrar por nivel pre-escolar para poder mostrar las matrículas de un nivel en específico. Esto se logra a través del método `filter()`, aplicado al state de matrículas en Vuex, el cual crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada.

La figuras 4.6, 4.7 y 4.8, son un buen ejemplo del diagrama de flujo para el formulario de creación, tabla y acciones de matrículas.

Mientras que la figura D.12, muestra el diagrama de flujo de la acción filtrar por nivel pre-escolar



Figura D.12: Diagrama de flujo de la acción filtra por nivel escolar en tabla matrículas

D.3.5. Receta

Desde el menú de navegación, se puede acceder a esta vista al hacer click en la opción Receta. En esta vista es posible crear, editar, copiar y eliminar recetas.

A continuación, se detallan los componentes que están en esta vista.

- **Formulario creación de receta:** Este formulario permite crear una nueva receta. Está compuesto de 3 elementos QSelect (Plato, Tipo colación y nivel pre-escolar) y existen la posibilidad de agregar todos los ingredientes (con su cantidad y unidad de medidas) necesarios para la creación de una receta..
- **Tabla de recetas:** Esta tabla está encargada de listar las recetas registradas en la sucursal en sesión.

Las acciones de esta tabla permiten filtrar por nivel pre-escolar, copiar, editar y eliminar la receta.

La figura D.13 muestra el diagrama de flujo de las nuevas acciones presentadas editar y copiar receta.

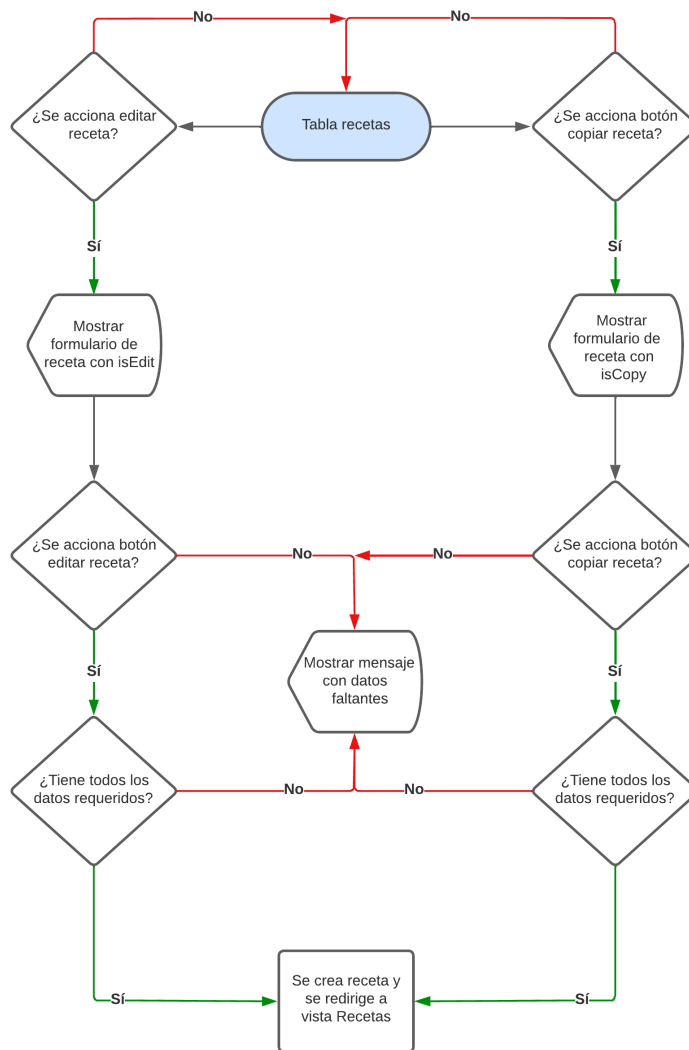


Figura D.13: Diagrama de flujo de la vista editar mis datos

D.3.6. Editar mis datos

Desde el menú de navegación, presionando la opción Editar mis datos, se llega a esta vista. Cuando se ingresa a esta vista, se realiza un Request a la API de tipo GET a la URL:

host/api/users/:userId

para obtener los datos del usuario que está en sesión. Cuando llega el Response, se cargan los datos del usuario en el mismo componente para crear un usuario. Cuando es apretado el botón EDITAR, se realiza otro Request a la API de tipo PUT a la misma URL que se mencionó anteriormente *host/api/users/:userId* para guardar los datos editados del usuario. En el body se adjunta el objeto del usuario editado. Al llegar el Response se muestra una notificación para mostrar que el usuario ha sido editado con éxito.

La figura D.14 muestra el diagrama de flujo de esta vista

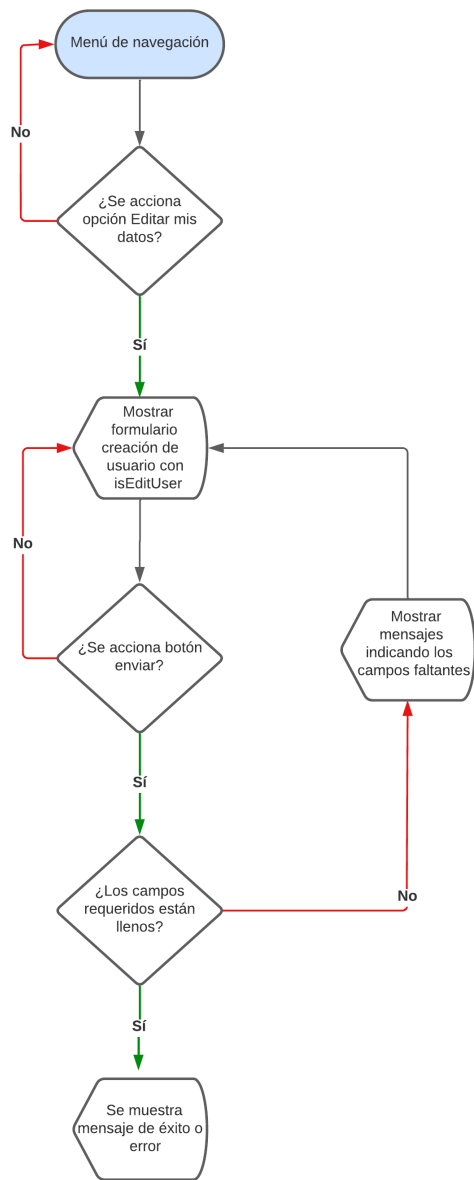


Figura D.14: Diagrama de flujo de la vista editar mis datos

D.3.7. Cerrar Sesión

En esta funcionalidad, al apretar el botón Cerrar Sesión, procede a eliminar los datos almacenados en el localStorage, para luego redirigir a la vista de Login. Esta acción se puede ver reflejada en el diagrama de flujo de la figura D.15

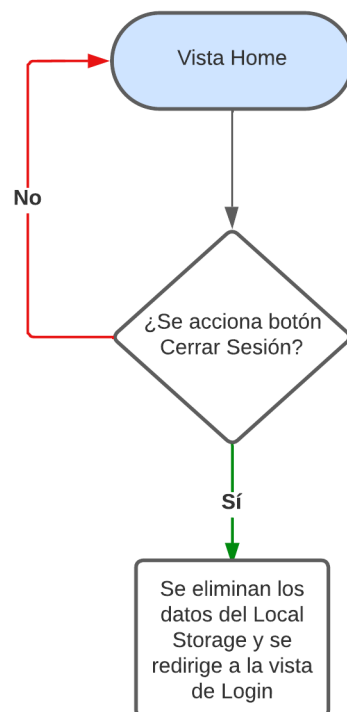


Figura D.15: Diagrama de flujo de la acción cerrar sesión

D.4. Imágenes de la aplicación

D.5. Vista Login

La figura D.17, es la vista inicial al momento de ingresar a intranet.natgio.cl. En el formulario de login, es necesario ingresar el usuario/email y la contraseña. Dependiendo del rol del usuario, se pasará a la siguiente vista, según sea Administrador (Selección de sucursal), Manipulador y Nutricionista (Home). Para dar identidad e institucionalidad a la plataforma, en el Login se agrega el logo del Jardín y Sala Cuna Natgio.

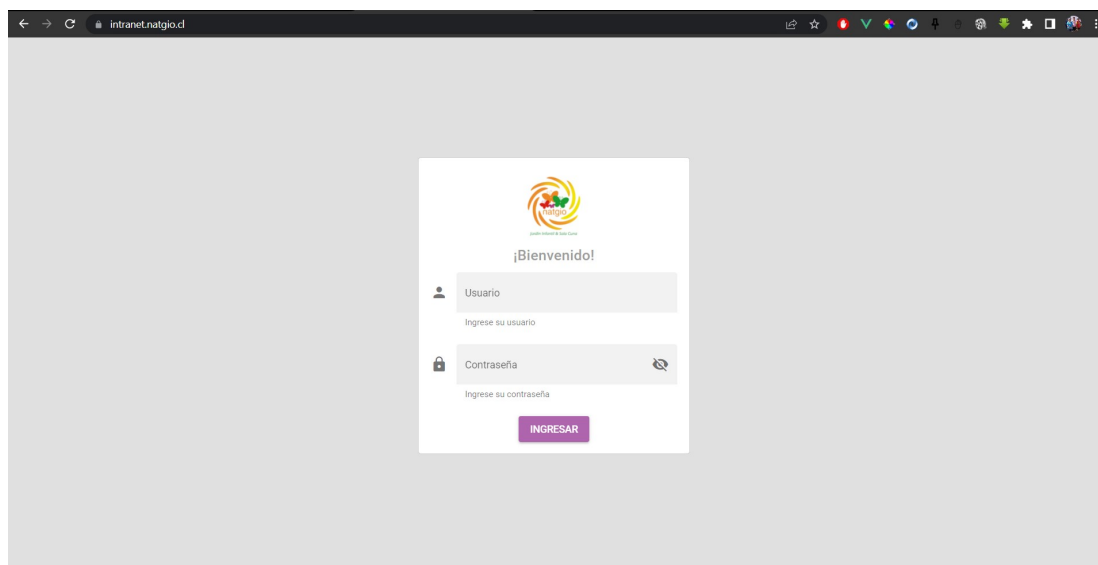


Figura D.16: Vista del Login.

D.6. Selección de sucursal

Si el usuario en sesión es un administrador, es necesario seleccionar la sucursal que estará en sesión. En esta vista, se listan todas las sucursales que están registradas en la base de datos. Hasta el momento, solo existe una.

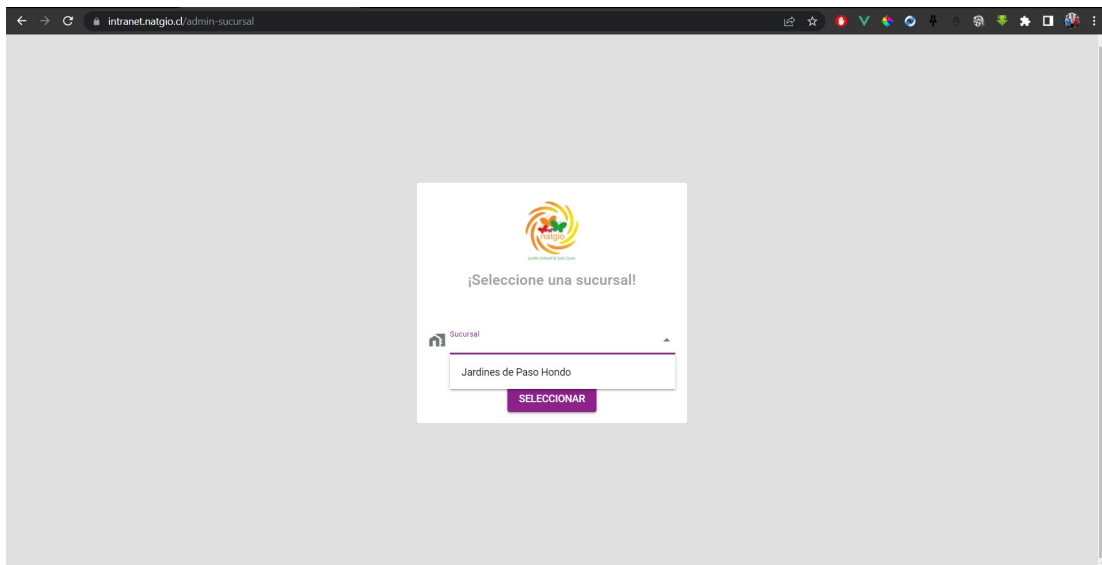


Figura D.17: Vista selección de sucursal.

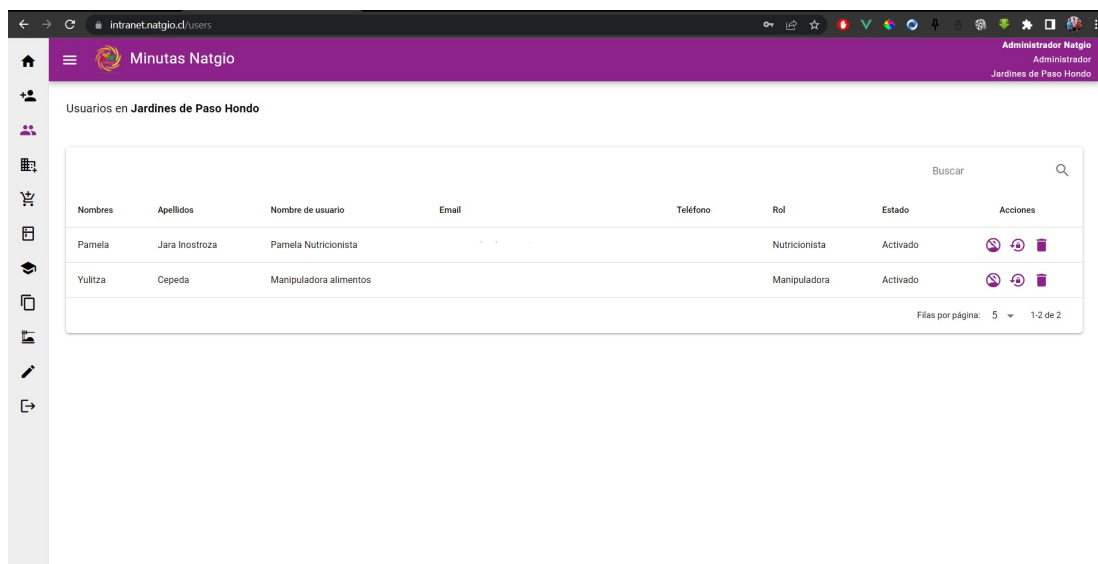
D.7. Vista Registrar

Desde el menú de navegación, es posible ingresar a la vista de la figura D.18, donde se pueden crear usuarios en la sucursal que está en sesión.

Figura D.18: Vista Registro de usuario en sucursal.

D.8. Vista Listar usuarios

Desde el menú de navegación, es posible ingresar a la vista de la figura D.19, donde se listan todos los usuarios de la sucursal que está en sesión. Se han borrado los correos y teléfonos de los usuarios que están registrados, para resguardar la información de ellos. Además, en esta figura es posible ver las acciones activar/desactivar usuario, cambiar contraseña y eliminar usuario.



Administrador Natgio
Administrador
Jardines de Paso Hondo

Usuarios en Jardines de Paso Hondo

Buscar

Nombres	Apellidos	Nombre de usuario	Email	Teléfono	Rol	Estado	Acciones
Pamela	Jara Inostroza	Pamela Nutricionista			Nutricionista	Activado	  
Yulitza	Cepeda	Manipuladora alimentos			Manipuladora	Activado	  

Filas por página: 5 1-2 de 2

Figura D.19: Vista Registro de usuario en sucursal.

D.9. Vista Creación variables institucionales

A través del menú de navegación, se accede a la vista de la figura D.20. En esta vista, se pueden crear sucursales y otros niveles pre-escolares. Además, la figura muestra cómo se puede editar el nombre de la variable.

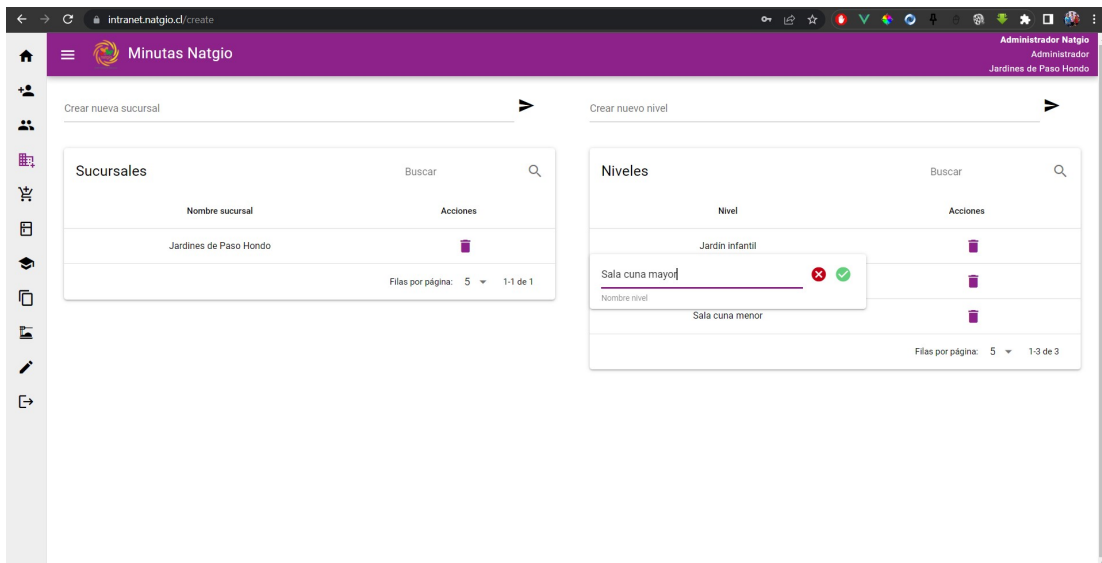


Figura D.20: Vista Creación variables institucionales

D.10. Vista Creación variables alimentarias

A través del menú de navegación, se accede a la vista de la figura D.21. En esta vista, se pueden crear tipos de colaciones, platos e ingredientes. La figura D.22, muestra el modal de confirmación para eliminar un dato de alguna tabla.

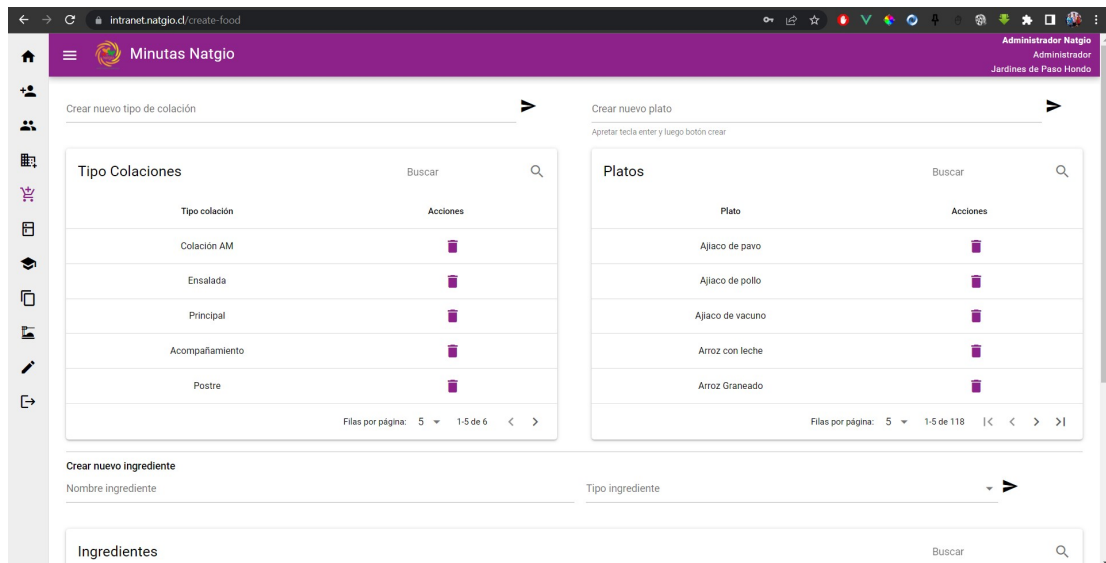


Figura D.21: Vista Creación variables alimentarias

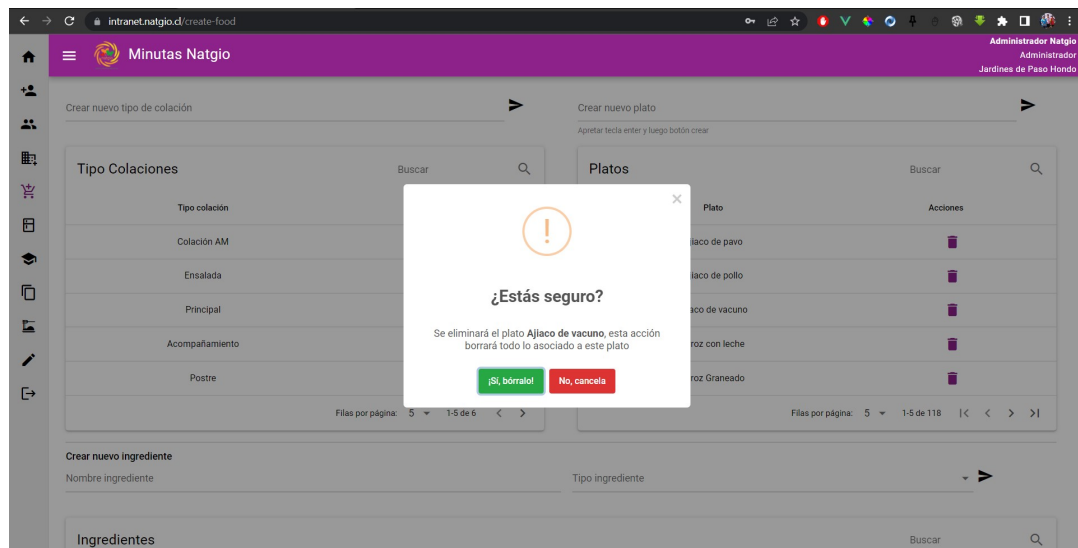


Figura D.22: Modal confirmación de eliminación de dato

D.11. Vista Stock

Esta vista, es accesible desde el menú de navegación. En la figura D.23, se pueden observar las dos tablas de ingredientes con su respectivo stock en la sucursal en sesión.

Stock en Jardines de Paso Hondo

Ingrediente	Stock	Unidad de medida
Aceite	0	Centrmetro cúbico [cc]
Agua	0	Centrmetro cúbico [cc]
Arroz	0	Gramo [g]
Atún	0	Gramo [g]
Avena	0	Gramo [g]
Barra cereal	0	Unidad
Canela	0	Gramo [g]
Cereales	0	Gramo [g]
Chocolo congelado	0	Gramo [g]
Condimentos	0	Gramo [g]
Crema	0	Gramo [g]
Endulzante	0	Gotas
Fideos	0	Gramo [g]
Flan instantáneo	0	Gramo [g]
Fórmula Láctea	0	Gramo [g]
Galleta María	0	Unidad
Galletas Integrales	0	Unidad
Garbanzos legumbre	0	Gramo [g]

Ingrediente	Stock	Unidad de medida
Acelga	0	Gramo [g]
Albahaca	0	Gramo [g]
Betarraga	0	Gramo [g]
Brócoli	0	Gramo [g]
Cebolla	0	Gramo [g]
Fruta	0	Gramo [g]
Lechuga	0	Gramo [g]
Papas	0	Gramo [g]
Pepino	0	Gramo [g]
Repollo	0	Gramo [g]
Tomate	0	Gramo [g]
Zanahorias	0	Gramo [g]
Zapallo	0	Gramo [g]
Zapallo italiano	0	Gramo [g]

Filas por página: 14 1-14 de 14

Figura D.23: Vista Stock

D.12. Vista Matrículas

Esta vista, es accesible desde el menú de navegación. En la figura D.24, se muestra el formulario para generar las matrículas en un mes y año específico para todos los niveles y tipos de colaciones en la sucursal en sesión.

Crear matrícula en Jardines de Paso Hondo

Mes: Año: GENERAR MATRÍCULA

Matriculas Jardín infantil Sala cuna mayor Sala cuna menor Todos Buscar

Nivel escolar	Mes	Año	Número de estudiantes	Tipo colación	Acciones
Jardín infantil	Noviembre	2022	52	Colación AM	
Sala cuna mayor	Noviembre	2022	13	Colación AM	
Sala cuna menor	Noviembre	2022	3	Colación AM	
Jardín infantil	Noviembre	2022	52	Ensalada	
Sala cuna mayor	Noviembre	2022	13	Ensalada	
Sala cuna menor	Noviembre	2022	3	Ensalada	

Filas por página: 6 1-6 de 18

Figura D.24: Vista Matriculas

D.13. Vista Copiar minutas

En la figura D.25, se muestra la vista que permite copiar minutas en la sucursal que está en sesión. Se puede observar el formulario que permite esta acción.

Selecciona nivel y tipo de colación a copiar

Nivel Colación

Selecciona los días de la semana a copiar

Selecciona el primer día del rango a pegar

COPIAR MINUTAS

Figura D.25: Vista Copiar minutas

D.14. Vista Recetas

En esta vista, es posible crear, copiar, editar y eliminar recetas. En la figura D.26, se muestra el formulario para crear receta y la tabla con todas las recetas creadas en la sucursal. Mientras que la figura D.27, muestra la vista para editar la receta. La vista para copiar una receta es similar.

The screenshot shows the 'Vista Recetas' (Recipe View) in the Minutas Natglo system. The interface is in Spanish and shows a form for creating a new recipe and a table of existing recipes.

Formulario para crear receta:

- Crear receta en Jardines de Paso Hondo**
- Plato:** [Dropdown menu]
- Tipo Colación:** [Dropdown menu]
- Nivel escolar:** [Dropdown menu]
- Ingrediente:** [Dropdown menu]
- Cantidad:** [Input field]
- Unidad de medida:** [Dropdown menu]
- Para valores decimales ocupar punto**
- + AGREGAR INGREDIENTE**
- CREAR RECETA** (Botón verde)

Mis recetas:

Legend: ● Jardín infantil ● Sala cuna mayor ● Sala cuna menor ● Todos

Buscar [Input field]

Nivel	Plato	Tipo de colación	Ingredientes	Acciones
Jardín infantil	Arroz Graneado	Acompañamiento	Ver ingredientes	[Icono de copiar] [Icono de editar] [Icono de eliminar]
Sala cuna mayor	Arroz Graneado		Ver ingredientes	[Icono de copiar] [Icono de editar] [Icono de eliminar]

Figura D.26: Vista Recetas

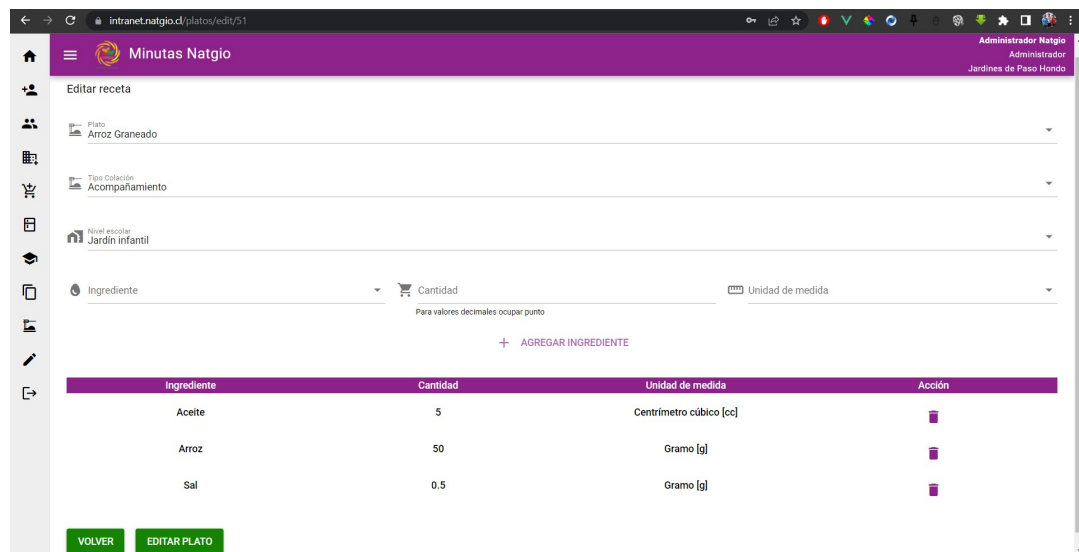


Figura D.27: Vista Editar Receta

D.15. Vista Editar mis datos

Al igual que las vistas anteriores, se accede a través del menú de navegación. En esta vista, es posible editar los datos del usuario que está en sesión. La figura D.28, muestra los datos del usuario en sesión.

The screenshot shows a web browser window with the URL `intranet.natgio.cl/edit`. The page title is "Minutas Natgio". The user is logged in as "Administrador Natgio" (Administrator) with the role "Administrador" and the branch "Jardines de Paso Hondo".

The form is titled "Editar mis datos" (Edit my data) and contains the following fields:

- Nombre** (Name): Administrador Natgio
- Apellidos** (Surname): Sanhueza Mejias
- Correo electrónico** (Email): javier@email.cl
- Nombre de usuario** (Username): administrador
Nombre de usuario debe tener mínimo 6 caracteres
- Número de teléfono** (Phone number): +56 9 8855 0667
- Ingrese una contraseña** (Enter a password): *****
Contraseña debe tener mínimo 6 caracteres
- Rol del usuario** (User role): Administrador
- Sucursal** (Branch): Jardines de Paso Hondo

Figura D.28: Vista Editar mis datos

Bibliografía

- [1] Danny Sapio. 9 material design guidelines, according to google. <https://buildtin.com/design-ux/10-insights-googles-material-design-guidelines>, Octubre 2022.
- [2] Marcos Mella Javiera García, Héctor Arraigada. Nutrigreen. <https://www.usach.cl/news/software-automatiza-diseno-minutas-para-casinos-alimentacion-gana-version-2021-del-concurso>, 2021.
- [3] Ch. Sai Kiran K. Sai Pavan B. Bhumika V. Sateesh Abdul Rahaman, Viyyapu Gayatri. Development of web applications by integrating frontend and backend tools. <http://junikhyatjournal.in/>, 2022.
- [4] Juan De Assembler Institute. ¿cuántos lenguajes de programación existen? <https://assemblerinstitute.com/blog/blog-cuantos-lenguajes-de-programacion-hay>, 2022.
- [5] Stack Overflow. Encuesta stack overflow 2022. <https://survey.stackoverflow.co/2022/>, 2022.
- [6] Tamara Pattinson. Relational vs. non-relational databases. <https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases>, Noviembre 2022.

- [7] Alex Phillips. What is a relational database? features uses. <https://www.salesforce.com/uk/blog/2022/01/what-is-a-relational-database.html>, Enero 2022.
- [8] Google. Calendario de google. <https://workspace.google.com/intl/es-419/products/calendar/>.
- [9] Ivan de Souza. Archivo json: ¿qué es y para qué sirve en las páginas web? <https://rockcontent.com/es/blog/archivo-json/>, Agosto 2021.