**UNIVERSIDAD TECNICA FEDERICO SANTA MARIA**

**Repositorio Digital USM** **https://repositorio.usm.cl**

Tesis USM TESIS de Pregrado de acceso ABIERTO

2021-11

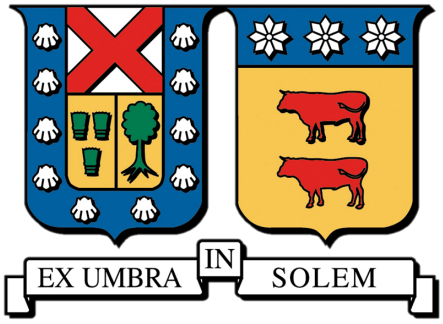
# PREDICCIÓN DE DEMANDA DE AGUA POTABLE MEDIANTE APRENDIZAJE DE MÁQUINAS Y ANÁLISIS DE SERIES DE TIEMPO

ARAYA BARROS, LUIS FRANCISCO

https://hdl.handle.net/11673/52766

*Repositorio Digital USM, UNIVERSIDAD TECNICA FEDERICO SANTA MARIA*

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**
**DEPARTAMENTO DE ELECTRÓNICA**
**VALPARAÍSO - CHILE**




# "PREDICCIÓN DE DEMANDA DE AGUA POTABLE MEDIANTE APRENDIZAJE DE MÁQUINAS Y ANÁLISIS DE SERIES DE TIEMPO"

**LUIS FRANCISCO ARAYA BARROS**
**MARCELO IVÁN GONZÁLEZ HENRÍQUEZ**

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

**PROFESOR GUÍA1:** **MOHAMED ABDELHAMID**
**PROFESOR GUÍA2:** **NICOLÁS JARA**

**Noviembre 2021**

# Agradecimientos Luis

Dedicado a todos los compas conocidos en mi paso por la USM, amigos del basket y pichangas, tías y tíos del aseo y comedor, Flake, Anderson y Carlos Ario. Y en especial a toda la carrera de telemática tanto alumnos como administrativos.

Telemática lo + grande, campeones intercarreras basket 2019.

# Agradecimientos Marcelo

Quiero agradecer a mi familia por su apoyo incondicional en las decisiones que he tomado y ser los pilares de mi desarrollo como persona.

A mis amig@s de Playa Ancha por todas las aventuras, los momentos incontables de risa y el apañe en tiempos complicados.

A *los cabros del Liceo*, que a pesar de que tomamos caminos distintos, han estado presentes de alguna forma en este proceso.

A amig@s de la Universidad con los que compartí estudiando o en carretes *tirando memes* y/o contando anécdotas.

A la comunidad Telemática, en donde cada uno de los miembros cooperó con un granito de arena en mi formación, tanto académica como personal. Gracias por poner a disposición espacios en donde pude cooperar a la comunidad y al mismo tiempo desarrollarme integralmente.

# Resumen

El agua como recurso es vital para el ser humano y además es una principal fuente de desarrollo económico y social del planeta. Factores como el aumento de la población, aceleración del cambio climático y nulas políticas de sostenibilidad han causado que dicho recurso sea cada vez sea más escaso, generando periodos extensos de sequía en varios países, incluyendo Chile.

Una de las industrias que se destacan en el uso de dicho recurso son los sistemas de distribución de agua, los cuales se encargan del tratamiento de aguas servidas, además de la producción y distribución de agua potable. En este sentido, es vital el uso óptimo de los recursos, minimizando factores de pérdida como roturas de las conexiones, rebalses y mala toma de decisiones. La operación reactiva de los operadores frente a escenarios inciertos causan mermas importantes de energía, recursos monetarios y de agua.

En esta tesis de título se trabaja con una compañía de servicios sanitarios ubicada en la Región de Valparaíso con el objetivo de minimizar las pérdidas asociadas a la toma de decisiones diarias de los operarios del sistema. En particular, se desea disponer de herramientas predictivas para que los operarios puedan tomar mejores decisiones en el corto plazo.

Se propone un sistema compuesto por un flujo que construye modelos de predicción capaces de estimar el consumo diario de agua en base al consumo histórico para las zonas más críticas consideradas por la empresa. Además, se incluye una aplicación web en donde los operarios pueden consultar sobre el consumo diario en cada zona de interés para así poder tomar decisiones de forma proactiva y en consecuencia una minimización de costos operacionales.

***keywords: despacho económico de carga, sistema de distribución de agua*** ***(SDA)***, ***forecasting, aplicación web, machine learning, series de tiempo.***

# Abstract

Water as a resource is vital for human beings and is also a main source of economic and social development. Population growth, climate change and lack of sustainability policies have caused to increase water scarce, generating extended periods of drought in several countries, including Chile.

Water distribution systems, which are responsible for the treatment of sewage, as well as the production and distribution of drinking water, stand out as the responsible to manage this resource. Reduce loss factors such as broken connections, overflows and poor decision making, is vital for the optimal use of the resource. Reactive operation in the face of uncertain scenarios causes significant losses of energy, monetary resources and water.

The objective of this thesis is to work with a sanitary service company located in the region of Valparaíso, Chile, in order to minimize the losses associated with the daily decision making of the system operators. In particular, it is desired to have predictive tools so that the operators can make better decisions in the short term.

We propose a system workflow that builds predictive models capable of estimating daily water consumption based on historical consumption for the most critical areas considered by the company. In addition, a web application is included where operators can consult on the daily consumption in each zone of interest in order to make proactive decisions and consequently minimize operational costs.

***keywords: economic dispatch, water distribution system*** **(*WDA*), *forecasting, web application, machine learning, time series.***

# Glosario

***Serie de tiempo***: Secuencia de datos u observaciones, medidos en determinados momentos y ordenados cronológicamente.
***Outlier***: En estadística, un outlier es una observación que es numéricamente distante del resto de los datos.
***CSV***: Tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.
***HTML***: Lenguaje de marcado de hipertexto, que le permite al usuario crear y estructurar secciones, párrafos, encabezados, enlaces y elementos de cita en bloque para páginas web y aplicaciones.
***JSX***: Es una extensión de JavaScript creada por Facebook para el uso con su librería React.
***MVC***: Es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.
***Python***: Lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código.
***WSGI***: Es una convención de llamada simple para que los servidores web reenvíen solicitudes a aplicaciones web o marcos escritos en el lenguaje de programación Python.
***Multiparadigma***: La Programación Multiparadigma es una práctica que emerge como resultado de la co-existencia de los paradigmas orientado a objetos, procedural, declarativo y funcional buscando mejorar la producción en el desarrollo de proyectos.
***SQL***: Es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
***NoSQL***: Es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR en aspectos importantes, siendo el más destacado que

no usan SQL como lenguaje principal de consultas.

***API***: Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción.

***ORM***: Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

***HTTP***: Protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web.

***API REST***: La transferencia de estado representacional o REST es un estilo de arquitectura de software para sistemas hipermedia distribuidos.

***JSON***: Es un formato de texto sencillo para el intercambio de datos.

***JWT***: Es un estándar abierto basado en JSON propuesto por IETF para la creación de tokens de acceso que permiten la propagación de identidad y privilegios.

# Acrónimos

**BD**: Base de Datos

**SISS**: Superintendencia de Servicios Sanitarios

**SDA**: Sistema de Distribución de Agua

**LP**: Programación Lineal

**ARIMA**: AutoRegressive Integrated Moving Average

**ML**: Machine Learning

**NaN**: Not a Number

**N/A**: Not Available

**RMSE**: Root Mean Square Error

**MAPE**: Mean Absolute Percentage Error

**NN**: Neuronal Network

**RNN**: Recurrent Neuronal Network

**QMMP+**: Qualitative Multi-Model Predictor Plus

**SARIMA**: Seasonal AutoRegressive Integrated Moving Average

**RF**: Random Forest

**LSTM**: Long Short-Term Memory

**VAR**: Vector AutoRegression

**ANN**: Artificial Neuronal Network

**DNN**: Deep Neuronal Network

**LSSVM**: Least Square Support Vector Machine

**GP**: Gaussian Process

**FFNN**: FeedForward Neuronal Network

**KNN**: K-Nearest Neighbors

**SVR**: Support Vector Machine

**MAD**: Median Absolute Deviation

**csv**: comma-separated values

**SPA**: Single Page Application

**HTML**: HyperText Markup Language

**MVC**: Model View Controller

**SQL**: Structured Query Language

**SGBD**: Sistema de Gestión de Bases de Datos

**API**: Application Programming Interface

**REST**: Representational State Transfer

**ORM**: Object Relation Mapper

**JSON**: JavaScript Object Notation

**JWT**: JSON Web Token

**HTTP**: Hypertext Transfer Protocol

# Índice de figuras

# Índice general

# Capítulo 1

# Introducción

El agua, como recurso, es indispensable para la vida y un punto clave para el desarrollo económico y social del mundo actual. El rápido aumento de la población y crecimento de ciudades, como también la mala gestión y casi nulas políticas de sustentabilidad, han generado grandes periodos de sequía en muchos países [1].

El uso de agua compete a diferentes sectores de usuarios como minería, agricultura, ganadería y habitacional [2]. El creciente valor económico y medio ambiental del agua ha propiciado una clara necesidad de innovación en la tecnificación de métodos y procesos en las empresas del sector sanitario. Se deben desarrollar metodologías destinadas a optimizar la asignación de agua potable logrando así el máximo beneficio para la sociedad, cuidando de preservar el medio ambiente en que está inserta y teniendo en cuenta que el agua potable constituye actualmente un bien que debe poseer ciertas características mínimas, no sólo en cuanto a calidad físico - química y bacteriológica, sino también un rango de presiones adecuado en la red, así como la sustentabilidad del bien del recurso.

En Chile, Esval [3] es una compañia dedicada a la producción y distribución de agua potable, como también a la recolección, descontaminación y disposición de aguas servidas a gran cantidad de comunas de la Región de Valparaíso. La mejora en la gestión del recurso es una de las prioridades de la compañia, y tiene directa

relación con las pérdidas por roturas, rebalses y mala toma de deciciones.

Las pérdidas, casi en su totalidad, están asociadas a múltiples factores, tales como: la operación de las redes de abastecimiento y distribución, su compleja topología, la variabilidad continua de la demanda, la cantidad de parámetros que deben considerarse para realizar un despacho económico que permita satisfacer sus requerimientos, las distancias intrínsecas en la propia red y la toma de deciciones apresuradas por parte de los operadores al enfrentarse a un escenario inesperado. Esto último, conlleva a un proceso de toma de decisiones propiciada por elementos subjetivos de la persona que opera el sistema y no por estándares que se basen en un suministro óptimo (operación reactiva).

El último factor descrito, operación reactiva, es lo que se desea solucionar con el desafío propuesto por Esval en el contexto del Programa de Memorias Multidisciplinarias de la Universidad Técnica Federico Santa María, año 2019.

Actualmente Esval cuenta con un sistema de monitoreo con el cual determinan, en tiempo real, el volumen de agua que disponen los estanques, el estado de válvulas de paso y el flujo de agua de las plantas productoras. El desafío contempla desarrollar un modelo predictor de demanda y un modelo de optimización, capaz de entregar un plan de acción diario, con el objetivo de reducir las pérdidas asociadas a la operación reactiva.

El presente estudio contempla el desarrollo del modelo de predicción de demanda a partir de información histórica, esta solución se implementa a través de una aplicación web llamada WHOME, lo cual ofrece una solución parcial al desafío propuesto dado que se enfoca en su totalidad en generar información precisa para abastecer al modelo de optimización.

## 1.1. Objetivos

### 1.1.1. Objetivo General

Desarrollar un sistema que permita predecir la demanda de agua potable de las próximas 24 horas en la zona del Gran Valparaíso. En conjunto con el modelo de optimización permitirá a los operadores de red establecer planes de trabajo con el fin de cumplir siempre la demanda usuaria y se reduzca la cantidad de agua potable no utilizada que se bombea a la red.

### 1.1.2. Objetivos Específicos

- Analizar y limpiar información histórica de los estanques de la red.
- Desarrollar modelos para predecir la demanda de agua de la red.
- Definir método para disponibilizar modelos de predicción.
- Diseñar y desarrollar aplicación para presentar resultados obtenidos.

## 1.2. Estructura

Este documento se estructura de la siguiente forma: en el capítulo 2, se explica el proceso de extracción, distribución y monitoreo de agua potable que posee Esval; en el capítulo 3 se presentan estudios sobre optimización del proceso de distribución, también se detallan métodos de predicción y se presentan estudios de predicción de agua potable; en el capítulo 4 se detalla el procedimiento para obtener el dataset de entrenamiento, se explican los métodos de predicción utilizados, se comparan las distintas tecnologías para desarrolar el sistema, además de diagramas, arquitectura y componentes que lo conforman; mientras que en el capítulo 5, se presentan los resultados de los distintos modelos y se describe el desarrollo final de la aplicación. Por último, en el capítulo 6, se mencionan las conclusiones y se detallan las posibles mejoras que se pueden llegar a implementar a la solución.

# Capítulo 2

# Marco Teórico

El agua potable, de forma generalizada, llega a los consumidores a través de un sistema sanitario que consiste en una enorme obra ingenieril con la función principal de distribuir y potabilizar agua proveniente de fuentes naturales [4]. Este sistema está compuesto por una serie de etapas que procesan el agua no potable y la distribuyen a cada uno de los estanques de abastecimiento, la Figura 2.1, muestra un sistema de agua potable básico.

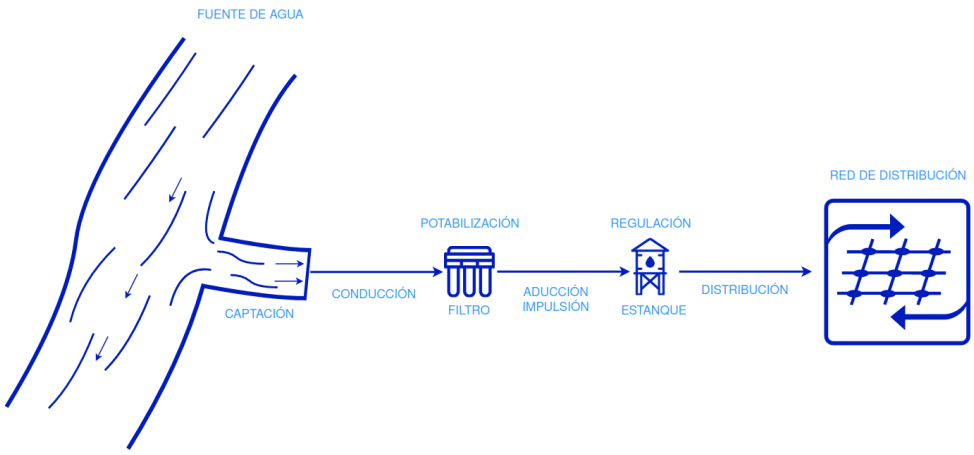



Figura 2.1: Sistema de agua potable. Fuente: Elaboración propia.

## 2.1. Proceso de extracción y distribución

Las funciones principales de este proceso son la captación, producción, regulación y distribución de agua potable, todos los procesos son regulados bajo los estándares de calidad establecidos por el Servicio Nacional de Salud [5]. La demanda de agua a analizar se limita a la zona geográfica del Gran Valparaíso, delimitada de color azul en la Figura 2.2.

### 2.1.1. Captación

La etapa de captación corresponde a la extracción y traslado de agua bruta desde una fuente natural, la cual puede tener distintos orígenes, hacia las plantas de producción. Algunos de los orígenes del agua bruta son; agua de lluvia almacenada, agua subterránea proveniente de manantiales, pozos o galerías filtrantes, agua superficial proveniente de ríos, arroyos o lagos, e incluso agua proveniente del mar (la cual requiere de procesos adicionales de desalinización).

El servicio de agua potable del Gran Valparaíso, está estructurado sobre la base de dos fuentes principales de abastecimiento; La primera es el río Aconcagua asociado con los sistemas productivos de agua potable de la planta Las Vegas y planta Concón, y la segunda fuente es el embalse natural de Peñuelas, donde se ubica la planta de tratamiento. Existen dos fuentes alternativas, embalse Poza Azul y embalse Los Aromos. El primero tiene un alto costo operativo, por lo que solo abastece una pequeña área del sector Quilpué, específicamente la zona urbana El Belloto. El embalse Los Aromos , se usa como apoyo potencial para la planta Concón, especialmente en verano debido a las variaciones hidrológicas del río Aconcagua.

### 2.1.2. Producción

La etapa de producción está subdividida por procesos específicos de potabilización, que corresponden a una serie de tratamientos de desinfección y filtración. Entre

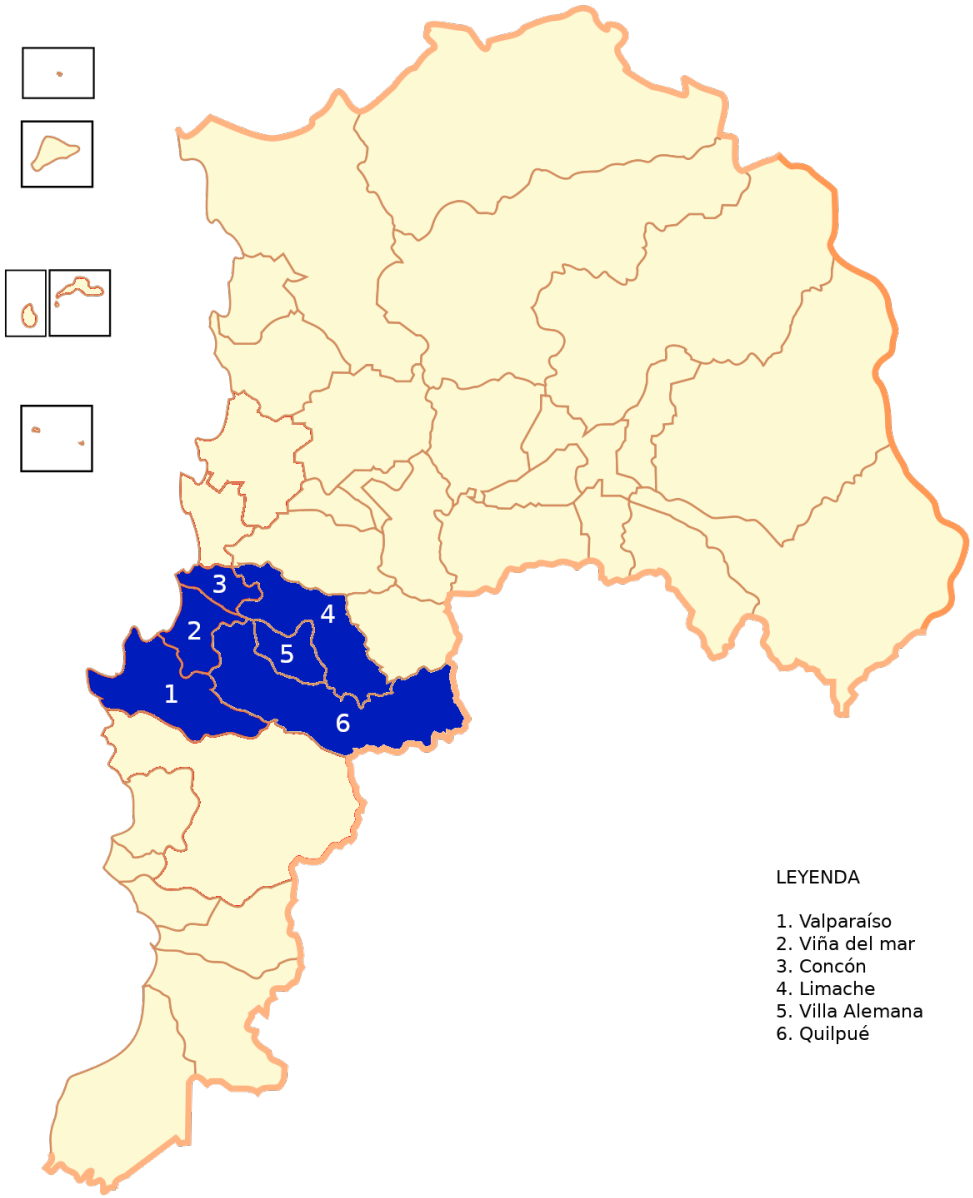



Figura 2.2: Mapa Región de Valparaíso.

otras cosas, el tratamiento de agua incluye etapas de retención de material grueso y de material fino en suspensión, tratamientos químicos de decantación de materiales muy finos y desinfección en general. Especificamente se utiliza gas cloro, sulfato de aluminio, flúor, hipoclorito de sodio, reactivos químicos, cloruro férrico, polímeros, carbón activado, polielectrolitos, soda cáustica y otros productos químicos, cumplien-

do así con la Norma Chilena Nch 409 de Calidad del Agua Potable [6] [7]. La Figura 2.3 describe el proceso de potabilización.


Figura 2.3: Etapas de Potabilización. Fuente: Reporte de sostenebilidad Esval 2020.

### 2.1.3. Regulación

La etapa de regulación está compuesta por una bomba de extracción que alimenta con agua tratada a un estanque de regulación (conocido popularmente como copa de agua), la cual tiene la función de compensar las variaciones horarias del consumo versus la frecuencia de funcionamiento de la bomba de extracción. Cabe notar, que el volumen de producción diario es igual al volumen de salida del estanque de regulación en 1 día. Lo anterior, se debe a que la bomba de extracción entrega agua a caudal constante por ciertos periodos de tiempo al día, donde el volumen total de agua bombeada al estanque corresponde a los litros necesarios para el abastecimiento de 1 día. Por lo tanto, la demanda diaria de consumo de agua es igual a la demanda de producción diaria. Adicionalmente, el estanque de regulación tiene la tarea de almacenar un volumen específico para casos de emergencia, de tal forma, de garantizar el abastecimiento de agua a los consumidores.

### 2.1.4. Distribución

La red de distribución consta de 4.666 kilómetros de cañerias subterráneas de agua distribuidas en toda la región de valparaíso [6]. Este sistema de cañerías llega a cada uno de los puntos de abastecimiento para los consumidores. La red se inicia en el estanque de regulación, y consta de estaciones de bombeo, tuberías, válvulas que sectorizan el suministro de agua en caso de rupturas y emergencias por escasez de agua, y, por último, de dispositivos de medición de volumen de agua en los puntos de abastecimiento (medidores).

## 2.2. Centro de monitoreo

Dada la complejidad del sistema de generación, mantención y distribución de agua potable en gran parte de la Región de Valparaíso, la empresa dispone de una Unidad Central de Distribución, la cual tiene la misión de recopilar de forma continua la información necesaria del estado del sistema para luego utilizarla en la toma de decisiones para cumplir el objetivo de impartir un buen servicio minimizando costos operacionales. Inicialmente, esta unidad fue operada por personal no especializado y solo con experiencia empírica. Recopilaban los datos periódicamente por teléfono con los operadores de las distintas plantas y estanques. Además, se registraban las decisiones tomadas en una planilla como por ejemplo el nivel de producción de cada planta o puesta en marcha de plantas elevadoras. Dada la falta de nivel técnico de los operadores, el costo de obtener datos y la calidad de éstos no era posible la optimización de los recursos.

En 1997 se puso en funcionamiento el Sistema de Telemetría, una red óptica desplegada por el Gran Valparaíso que permite supervisar los parámetros más característicos de la distribución y abastecimiento de agua potable.

Actualmente, la Unidad Central de Distribución es llamada Unidad de Telemetría, la cual controla la distribución de agua potable en el Gran Valparaíso las 24 horas del

día en todos los días del año. La unidad está conformada por operadores que trabajan en turnos continuos, manejando el sistema desde una base de operación. Si bien algunas operaciones se pueden aplicar de forma remota, pueden ocurrir procedimientos mal ejecutados o también inseguridad de la información recibida por parte del sistema de monitoreo, lo cual implica que las operaciones se lleven a cabo vía telefónica o radio con los operadores. La Unidad de Telemetría esta apoyada por operadores que cumplen la función de realizar las operaciones manuales y verificación del correcto funcionamiento de los distintos equipos de la red.

En esta unidad es donde se produce el problema planteado por Esval, ya que los operadores desarrollan su labor solo considerando el estado actual de la red y las decisiones que se realizan van en pos de siempre garantizar el recurso para los clientes. Con el fin de ayudar en la toma de decisiones en el centro de monitoreo, es que se propone generar planes de trabajo diarios para asistir a los operadores. Este plan de trabajo se genera a partir de un modelo de optimización que considera la demanda de agua potable futura, costo de las plantas de producción (considerando las etapas de elevación) y el nivel actual de los estanques de la red. En la Figura 2.4 se presenta el diagrama de la solución propuesta.

## 2.3. Aspectos comerciales

El negocio en el cual está involucrada la empresa corresponde a la obtención, distribución y comercialización de agua potable, y disposición de aguas servidas en la Región de Valparaíso. La gestión de recursos para permitir el suministro de agua potable a gran parte de la región es de real importancia y a la vez un trabajo complejo. En las diversas tareas que involucra ello, hay varias decisiones que son tomadas por los operadores en base a experiencias previas o bien observaciones subjetivas, careciendo de métricas para tomar estas. Lo anterior permite una entrega eficaz de agua potable a los clientes, pero no de forma eficiente. Factores como la complejidad de la topología de la red, la variación de la demanda en el tiempo, la cantidad

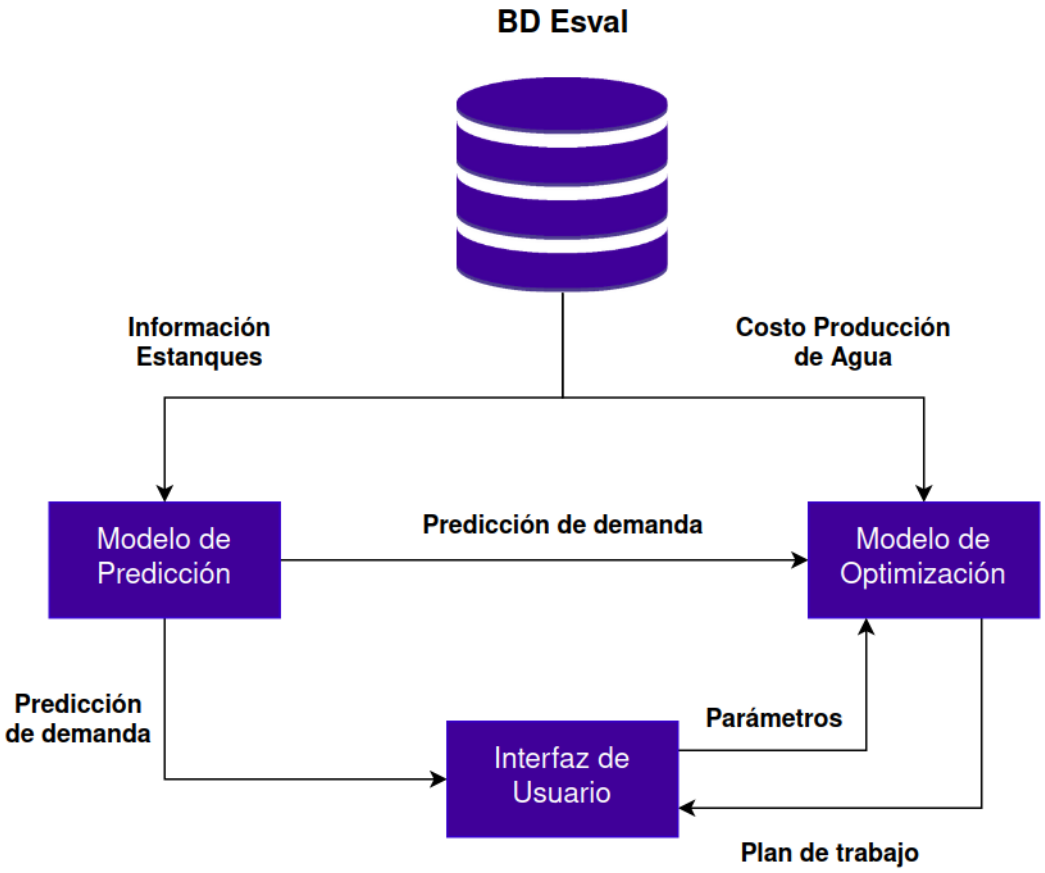



Figura 2.4: Esquema general del sistema. Fuente: Elaboración propia.

de parámetros que deben ser considerados por cada elemento de la red para ofrecer un despacho económico de carga que permita satisfacer la demanda y una infraestructura tecnológica limitada ocupada por los operadores causan que los operadores tomen decisiones subjetivas, sin considerar el uso óptimo de recursos.

Una de las mayores problemáticas que se enfrenta la empresa es la pérdida de agua potable en la operación de la red, tanto para abastecimiento como distribución, la cual es caracterizada como Agua No Facturada. Según la Superintendencia de Servicios Sanitarios (SISS) [8], el porcentaje de Agua No Facturada en el año 2018 de ESVAL S.A fue de 35,5 % mientras que a nivel nacional fue de 33.8 %. Dichas pérdidas tienen diversas causas, de las cuales destacan:

- Conexiones clandestinas y fraudes, mediante la intervención de los usuarios en la red de suministro.
- Roturas y filtraciones, debido al desgaste de las redes y variaciones bruscas de presión en éstas durante el día.
- Extinción de incendios.
- Error en mediciones.
- Mantenimiento de la red.

## 2.4. Requerimientos del sistema

Los requisitos funcionales definen una función del sistema o sus componentes, mientras que un requisito no funcional, define las restricciones o condiciones que impone el cliente, en este caso Esval. A continuación se definen los requisitos funcionales y no funcionales del sistema. Estos requisitos surgieron de las reuniones realizadas hasta la fecha, de manera que han sido validados por el cliente.

### 2.4.1. Funcionales

- El sistema debe entregar una predicción de la demanda de al menos 1 día.
- El sistema debe permitir cambiar el costo [$/lt] de cada planta.
- El sistema debe entregar como salida un plan de trabajo para cada planta.
- El modelo de predicción se debe "realimentar" al finalizar un día.
- El sistema debe contar con un gráfico de la predicción de demanda por hora.

### 2.4.2. No funcionales

- El sistema debe funcionar 24/7.
- El sistema debe contar con una interfaz intuitiva.
- El modelo de optimización debe ser rápido de ejecutar.

# Capítulo 3

# Estado del Arte

Los requerimientos de la vida moderna imponen nuevos desafíos a los administradores de sistemas de abastecimiento y distribución de agua potable. Los sistemas de distribución de agua, como infraestructuras críticas, están sujetos a picos de demanda debido a las fluctuaciones del consumo diario, como también a cambios a largo plazo en el patrón de demanda debido a la creciente urbanización. Para los proveedores del suministro, es imperante realizar un excelente diseño del sistema, con el fin de entregar el mejor servicio posible.

El problema combinatorio de un suministro de agua de alta calidad y, al mismo tiempo, de bajo costo, puede ser asistido por la optimización de la relación costo-beneficio para mejorar los sistemas de distribución existentes. La gestión del suministro de una ciudad es una labor compleja que depende fundamentalmente de la actuación de los operadores de la red de abastecimiento y distribución, y un soporte tecnológico que le facilite su gestión. En este capítulo, se exponen diferentes propuestas para optimizar estos procesos, haciendo especial énfasis en métodos de predicción de demanda de agua.

## 3.1. Redes de distribución de agua

Un sistema de distribución de agua (SDA) es un sistema que garantiza la distribución y el suministro de agua para satisfacer la demanda en diferentes zonas. Los SDA son sistemas que suelen contener módulos interrelacionados, como bombas, tuberías, válvulas, depósitos y tanques (ver Figura 3.1). Si bien, la gran cantidad de componentes de un SDA supone una operación compleja y muchos puntos de falla, también permite realizar mejoras, tanto a nivel de componentes específicos, como también a nivel de procesos que interconecten componentes.

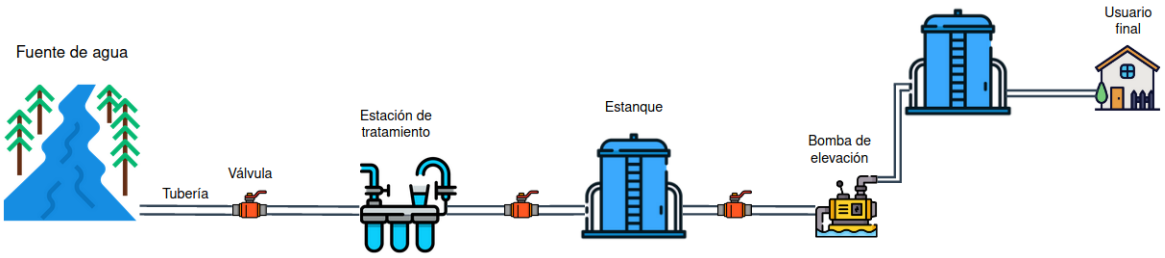



Figura 3.1: Diagrama general SDA. Fuente: Elaboración propia.

Uno de los trabajos estudiados [9], propone el concepto de estaciones de tratamiento y distribución de agua, y generaliza el sistema de suministro de agua en tres módulos: suministro de agua, estación de agua y usuario de agua. En donde a partir de un diagrama topológico de la red de agua, se establece un modelo refinado de asignación de recursos hídricos, la Figura 3.2 muestra el modelo propuesto. Tal modelo puede mostrar, en detalle, la fuente de suministro de agua, la cantidad de agua suministrada, la ingeniería de distribución de agua y otros datos de todos los usuarios de cada zona de distribución de agua. Esto permite realizar un análisis detallado de la oferta y la demanda de los usuarios, y proporcionar sugerencias y orientaciones teóricas para la implementación de la distribución regional del agua.

El modelo desarrollado en este trabajo es capaz de realizar una simulación refinada: los resultados de la simulación muestran con detalle los datos sobre el tipo de suministro y la cantidad de agua suministrada a cada usuario en cada zona. En

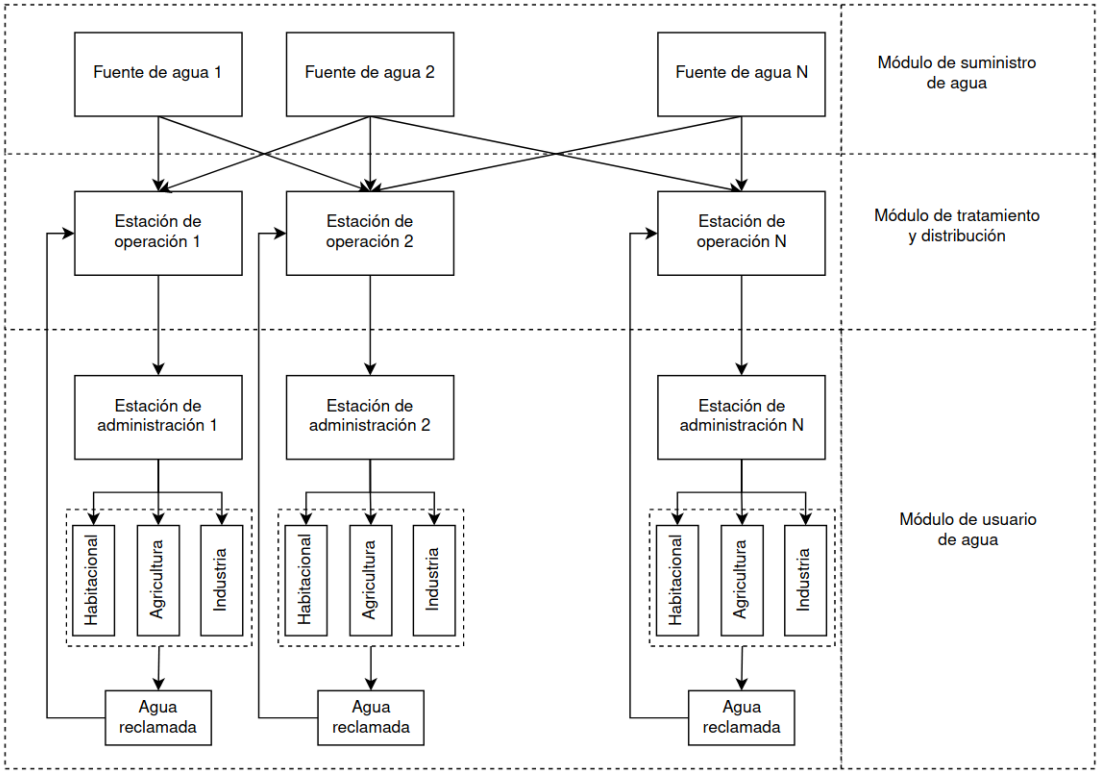



Figura 3.2: Modelo propuesto. Fuente: https://doi.org/10.3390/w10111580

consecuencia, se puede obtener información, como la dirección del suministro, la capacidad restante de la red de tuberías y la escasez de agua de los usuarios, mediante estadísticas. El mayor problema del modelo planteado es la gran cantidad de datos de entrada que necesita, además de requerir un amplio sistema de monitoreo de datos, como la demanda de agua de los usuarios, la red de suministro y su capacidad, y la capacidad de tratamiento de las plantas de agua. Por lo tanto, el modelo es una opción más adecuada cuando una red de distribución tiene datos detallados de su operación, pero tiene dificultades para proporcionar decisiones refinadas cuando una zona carece de datos suficientes.

Como se mencionó anteriormente, Esval cuenta con distintas fuentes de suministro de agua. Considerando la existencia de variadas fuentes de suministro, se deben tener en cuenta muchos factores en la toma de decisiones sobre el SDA, como la se-

lección de la fuente de suministro, los costos económicos, la calidad del agua y los requerimientos de la demanda. Estos factores no son independientes unos de otros y se deben equilibrar en el proceso de optimización de la asignación de los recursos hídricos, con el fin de llevar una operación eficiente y eficaz. En este ámbito, [10] presenta un modelo desarrollado y aplicado para servir como sistema de apoyo a la decisión multisectorial de suministro de agua para la gestión de los recursos hídricos teniendo en cuenta factores económicos y socio-ambientales. Se propuso un método de gestión de optimización multiobjetivo y multietapa para apoyar el uso combinado de aguas superficiales y subterráneas, también consideró el uso combinado de aguas superficiales locales, aguas subterráneas y fuentes de agua externas en la predicción de la demanda de agua. Específicamente se propone un modelo de programación lineal (LP) para ayudar a los responsables de la toma de decisiones en la planificación y el establecimiento de políticas para la asignación óptima de los recursos teniendo en cuenta los factores anteriormente nombrados.

El modelo fue aplicado en el área metropolitana de Beirut [11] para determinar el patrón de asignación multisectorial del agua que proporciona el mayor rendimiento neto por encima del uso del agua, al tiempo que se cumplen las principales limitaciones de disponibilidad de agua y las necesidades estacionales de agua per cápita. La comparación de los resultados de varios escenarios muestra que el rendimiento neto óptimo del uso del agua y la correspondiente asignación óptima del agua entre los distintos sectores varían considerablemente al introducir factores socio-ambientales. Al considerar estos factores el plan de asignación de recursos es limitado por los responsables de gestionar el recurso, que, mediante la aplicación de políticas en función de objetivos económicos, sociales o medioambientales específicos, alteran en gran medida la planificación entregada por el modelo.

## 3.2. Modelos de predicción

Como se especificó en la sección anterior, un SDA debe garantizar la generación, distribución y suministro de agua para alguna zona en particular. Dado que el agua es un recurso limitado y cada vez es más utilizada por más personas, es necesario que deba ser distribuida eficientemente. Además, dicha eficiencia ofrece una disminución de los gastos operacionales.

La eficiencia involucra que se deba producir solo lo necesario, intentando minimizar al máximo las pérdidas o esfuerzos extras para proveer la cantidad de agua requerida. En este sentido, es útil conocer la demanda de agua en un SDA con el objetivo de estimar cuánto es lo que se debe trabajar para poder suplir la demanda necesaria. A priori, se puede determinar que el consumo a través del tiempo es variable y que también depende de varios factores cíclicos como por ejemplo: estación climática, fines de semana o bien días feriados.

Dadas las condiciones del problema, se puede determinar que se desea predecir el consumo de agua en base a una serie de tiempo dada para los estanques de interés en la red. En otras palabras, se desea construir un modelo de predicción en donde el tiempo cobra relevancia. Hay dos enfoques principales para construir dicho modelo:

### 3.2.1. Análisis de series de tiempo

Es un campo de estudio basado en estadística que permite obtener predicciones considerando la dinámica de la serie de tiempo en el pasado. A partir de una serie de tiempo con $m$ muestras $\boldsymbol{X} = \left\{X_t\right\}_{t=1}^{m}$ y como variable de estudio $X_j$, se desea construir una función $f : \mathbb{R} \to \mathbb{R}$ tal que $y = f(t_j)$ sea la predicción de una variable de estudio en un tiempo $j, j > m$. Es posible categorizar los modelos de predicción de series de tiempo en dos tipos:

- **Modelos deterministas:** Son modelos de extrapolación simples en donde no se considera la naturaleza de la aleatoriedad de la serie. Al ser modelos sencillos,

tienden a tener mayor error de precisión.

- **Modelos estocásticos:** Son modelos que consideran que la serie $\boldsymbol{X}$ es construida a partir de un grupo de variables aleatorias con una cierta distribución conjunta difícil de obtener y por ende se utilizan modelos aproximados más simples que permitan obtener predicciones.

Otra característica importante de las series de tiempo es su estacionariedad:

- **Serie estacionaria:** Las series estacionarias se caracterizan por mantener constante su media y varianza en el tiempo, lo cual permite modelar con mayor precisión la dinámica de ésta.
- **Serie no estacionaria:** Dichas series se caracterizan por el cambio de la media y varianza a través del tiempo, dificultando su modelamiento.

El modelamiento de las series de tiempo tienen distintos enfoques:

- **Enfoque clásico:** En este enfoque se dice que una serie de tiempo puede ser descompuesta en cuatro componentes no observables y que solo se pueden obtener estimaciones. Dichos componentes, en función del tiempo $t$, corresponden a:
  - **Tendencia** ($T_t$): Refleja la progresión a largo plazo de la serie. Existe una tendencia cuando hay un aumento o disminución de la variable de interés que se mantiene en el tiempo. Es definida también como el cambio de la media a lo largo de un periodo extenso de tiempo.
  - **Ciclo** ($C_t$): Refleja las fluctuaciones repetidas pero no periódicas de la serie.
  - **Estacionalidad** ($E_t$): Refleja la variación estacional de la serie. Existe un patrón estacional cuando la serie está influenciada por factores estacionales, donde estos últimos ocurren en un periodo determinado, como por ejemplo, las estaciones climáticas o trimestres.

- **Aleatorio** ($A_t$): Refleja los movimientos erráticos que no siguen un patrón definido, producido por diversas causas. Representa los residuos o el resto de la serie temporal después que se hayan eliminado los demás componentes.

Un modelo clásico supone que la serie $\boldsymbol{X}$ puede ser descrita como la suma o el producto de sus componentes, es decir:

- **Modelo aditivo**: Se tiene que $X_t = T_t + C_t + E_t + A_t$.
- **Modelo multiplicativo**: Se tiene que $X_t = T_t \cdot C_t \cdot E_t \cdot A_t$.

- **Enfoque moderno:** En este enfoque se utilizan los modelos de Box-Jenkins que constan de 3 pasos:

  - **Identificación del modelo:** Se determina si la serie es estacionaria y si hay alguna estacionalidad significativa que deba modelarse. En base a lo anterior, se determina la naturaleza del modelo.
  - **Estimación del modelo:** Se utilizan algoritmos computacionales para obtener los coeficientes que mejor se ajustan al modelo seleccionado. Los métodos más comunes utilizan la estimación de máxima verosimilitud o la estimación por mínimos cuadrados no lineales.
  - **Validación:** Se comprueba si acaso el error a través del tiempo sigue los supuestos de un proceso estocástico univariado y estacionario, particularmente, los errores entre distintos instantes de tiempo deben ser independientes y además, la varianza y media de dicho proceso debe ser constante. Si se comprueba que no se cumple, se vuelve a la etapa de identificación para encontrar un modelo con mejor desempeño.

  Usualmente se les llama modelos ARIMA (AutoRegressive Integrated Moving Average), que es un modelo más concreto que siguen los pasos anteriormente descritos.

### 3.2.2. Machine Learning

Un algoritmo de aprendizaje automático (ML: Machine Learning) es un algoritmo que puede aprender de los datos. En [12] se propone una definición más precisa: "*Se dice que un programa de computadora aprende de la experiencia $E$ con respecto a alguna clase de tareas $T$ y medida de desempeño $P$, si su desempeño en las tareas de $T$, medido por $P$, mejora con la experiencia $E$*". En [13], se detalla dicha definición:

- **Tarea ($T$):** Corresponde a los tipos de tareas que los algoritmos de ML pueden resolver. Usualmente, las tareas son descritas en términos de cómo los algoritmos de ML debe procesar una muestra. Una **muestra** es una colección de características (features) que están representados cuantitativamente a partir de un objeto o evento que dichos algoritmos puedan procesar. Es representado por un vector $\boldsymbol{x} \in \mathbb{R}^n$ en donde cada componente $x_i, i \in \{1, ..., n\}$ corresponde a la feature $i$ del ejemplo, de un total de $n$ features. En el contexto del problema, un ejemplo puede ser una muestra en algún estanque de un SDA en un tiempo determinado que tenga la temperatura $(x_1)$ y caudal $(x_2)$. Las tareas más usuales de resolver con este tipo de algoritmos son:

  - **Clasificación:** En este tipo de tarea se busca que el algoritmo de ML asigne una clase a una muestra, de un conjunto de $k$ clases posibles. Para resolver la tarea, el algoritmo busca una función $f : \mathbb{R}^n \to \{1, ..., k\}$ tal que la evaluación de una muestra denotada por $y = f(\boldsymbol{x})$ corresponde a una categoría representada numéricamente, es decir, $y \in \{1, ..., k\}$.
  - **Regresión:** En las tareas de regresión, se busca que el algoritmo sea capaz de predecir $m$ valores numéricos en base a una muestra. Para resolver la tarea, el algoritmo busca una función $f : \mathbb{R}^n \to \mathbb{R}^m$ tal que la evaluación de una muestra denotada por $\boldsymbol{y} = f(\boldsymbol{x})$ corresponde a los $m$ valores predichos. Usualmente se desea predecir sólo un valor, lo que implica que $m = 1$ y la notación del resultado $\boldsymbol{y}$ sea reemplazada por $y$.

- **Detección de anomalías:** En este tipo de tarea se busca que el algoritmo agrupe muestras similares, tal que las muestras que no pertenezcan a ningún grupo sean clasificadas como anómalas.

- **Métrica de rendimiento** ($P$)**:** Para evaluar el desempeño de un modelo que provee los algoritmos de ML, es necesario establecer métricas cuantitativas. La elección de dichas métricas viene determinada por la tarea $T$ que se desea resolver. Por ejemplo, para las tareas de clasificación usualmente se utiliza la exactitud (accuracy) definida por la proporción entre la cantidad de muestras clasificadas correctamente y el total de muestras.

- **Experiencia** ($E$)**:** Los algoritmos de ML pueden ser categorizados a grandes rasgos por **supervisados** y **no supervisados**, dependiendo del tipo de experiencia que dichos algoritmos requieren durante el proceso de aprendizaje, entendiéndose por tipo de experiencia el conjunto de datos (dataset) que se utiliza para entrenar a los modelos de ML. Un dataset $\boldsymbol{X}$ puede ser representado por $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)}\}$, $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ en donde se tienen $m$ muestras con $n$ features cada una. Considerando lo anterior, el dataset debe ser organizado dependiendo de la categoría de los algoritmos de ML:

  - **Algoritmos supervisados:** Se utiliza un dataset con features, pero cada muestra se asocia a un valor llamado label o target. Por ejemplo, en una muestra de caudal en algún estanque en un instante dado, el tiempo es un feature mientras que la medición de caudal corresponde al target, si se desea predecir el caudal en un tiempo futuro. Principalmente, son utilizados para resolver tareas de regresión y clasificación.

  - **Algoritmos no supervisados:** Se utiliza un dataset con features pero a diferencia de los algoritmos supervisados, no cuenta con un valor target. En este tipo de algoritmos se aprenden propiedades útiles de la estructura del dataset con el fin de encontrar relaciones entre las mismas muestras. Son

utilizados para resolver tareas de detección de anomalías o clustering.

### 3.2.3. Proceso de entrenamiento

Antes de entrenar el modelo se deben preprocesar los datos, de modo que genere entradas y salidas adecuadas al modelo para lograr una buena estimación. En este proceso se normalizan valores, se eliminan datos atípicos (outliers), duplicados o sin información (N/A, NaN). En particular se utilizan sensores para medir parámetros de los estanques, los cuales pueden tener una mala calibración, o encontrarse inactivos y en consecuencia registrar datos inusuales.

Teniendo los datos históricos (preprocesados) de entrada y salida ($\boldsymbol{X}$ e $\boldsymbol{y}$), se necesitan dividir estos en porciones de entrenamiento y testeo: el primero es para entrenar el modelo, mientras que el segundo servirá para medir la exactitud del modelo bajo nuevas observaciones no incluidas en la porción de entrenamiento. El proceso se puede resumir según la Figura 3.3.

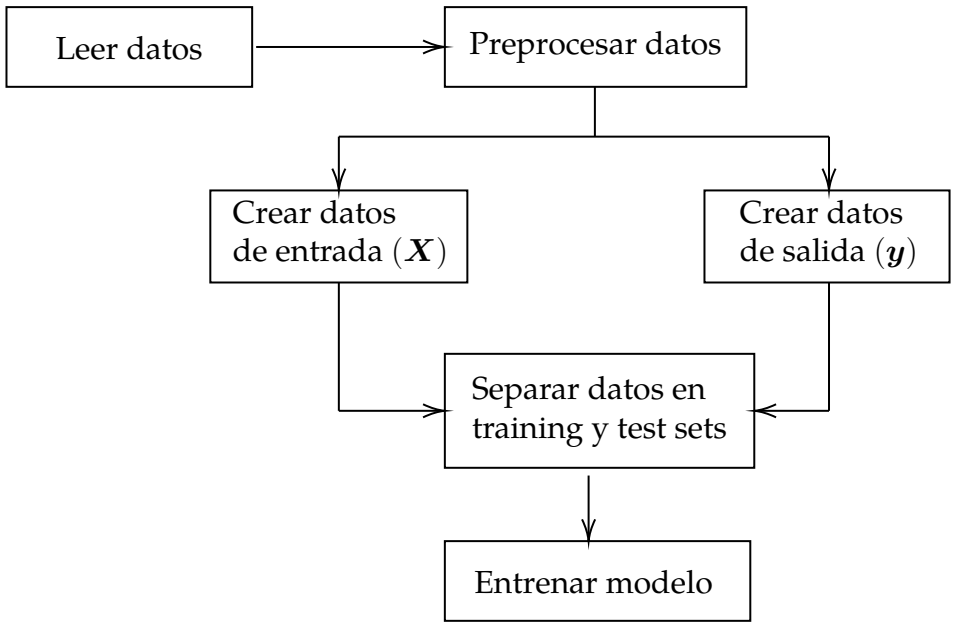



Figura 3.3: Diagrama del proceso de entrenamiento. Fuente: Elaboración propia.

### 3.2.4. Evaluación del rendimiento

Hay distintas métricas para cuantificar el error de predicción en modelos de regresión. Las más utilizadas son:

- **Root mean square error (RMSE)**:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

- **Mean absolute percentage error (MAPE)**

$$MAPE = \frac{100\,\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{3.2}$$

donde $n$ es la cantidad de datos, $y_i$ e $\hat{y}_i$ es el valor real y el valor predicho para el dato $i$, respectivamente. Es necesario destacar que dichas métricas son aplicables tanto para los algoritmos de ML como los modelos basados en análisis de series de tiempo.

Por otro lado, es necesario medir la capacidad de predicción de los modelos. En concreto, se debe estudiar el desempeño del modelo en base a datos que éste no ha observado. En este sentido se pueden dar dos fenómenos:

- **Subajuste:** En dicho fenómeno el modelo entrenado no se ajusta adecuadamente al test set, por lo que tiene un error alto en dicho set.

- **Sobreajuste:** En dicho fenómeno el modelo entrenado se ajusta adecuadamente al set de entrenamiento pero tiene un alto error para predecir valores utilizando el test set. En consecuencia, el modelo carece de capacidades predictivas.

## 3.3. Trabajos relacionados

En la literatura existen casos de estudio de predicción de demanda para SDA. En [14], se desarrollan modelos de predicción basados en la heurística de Redes Neuronales (NN) que puedan apoyar a la toma de decisiones operacionales junto con comparar éstos con un modelo más tradicional como la regresión lineal, considerando variables que pueden afectar a la demanda como la temperatura, precipitaciones y nieve. Además, se estudia el impacto que tienen dichas variables en las predicciones. Para cada modelo propuesto, se considera tanto una escala de tiempo mensual como diaria ya que la primera es útil para efectos de planificación mientras que la segunda son necesarias para apoyar en la toma de decisiones operacionales diarias. Para diseñar y validar los modelos propuestos por los autores, se ocuparon datos de la región de Central Indiana (Estados Unidos) entre los años 1997-2016 en donde se considera el consumo diario, ingresos familiar promedio, fecha, feriados, fines de semana, temperatura mínima y máxima, precipitaciones y nieve. Al analizar los modelos propuestos, se concluye que un modelo basado en redes neuronales recurrentes (RNN) entrega mejores resultados, tanto en la estimación diaria como mensual. Además, se determina que las variables con mayor relevancia son la temperatura máxima, el día del año y las precipitaciones: considerando sólo ellas, el modelo basado en RNN mejora su desempeño.

Por otro lado, en [15] se propone el modelo llamado Qualitative Multi-Model Predictor Plus (QMMP+) el cual se basa en estimar el consumo diario de agua a partir de análisis de series de tiempo de datos históricos en términos cuantitativos y cualitativos asumiendo que la serie de tiempo es cíclica, lo cual es recurrente en el dominio del problema. En particular, QMMP+ utiliza SARIMA para la cuantificación, mientras que se ocupa un algoritmo de clasificación basado en Nearest Neighbors junto con información de los días de la semana para estimar el patrón de consumo, que corresponde a comportamientos cualitativos. Para validar el modelo propuesto, se utiliza los datos de consumo de un SDA ubicado en Barcelona, España en el año

2012 y se compara el desempeño de QMMP+ con otros modelos de regresión como SARIMA, Naive Bayes y NN, demostrando que QMMP+ tiene mejor desempeño.

En [16], se prueban modelos basados en ML como NN, Random Forest (RF), Support Vector Machines, K-Nearest Neighbors para obtener predicciones a corto plazo en donde estos son evaluados con un set de datos de dos empresas del rubro en Portugal. Además, se analiza el impacto de factores como el clima, temporada, cantidad de datos utilizados en la fase de entrenamiento y ventana de tiempo de la predicción. Se sugiere una estrategia paralela ponderada que considera las ventajas de las diferentes técnicas probadas, de forma que la predicción esté compuesta por una suma ponderada de las predicciones de los modelos que mejor desempeño tengan en términos de error. Se obtuvo que las metodologías de ML, en la mayoria de las pruebas supera en desempeño al modelo ARIMA, además de concluir que es viable utilizar la estrategia planteada, asumiendo que los modelos a ocupar satisfacen un conjunto de prerrequisitos en relación a su rendimiento esperado.

En los últimos años se han conseguido mejoras significativas en la predicción de series de tiempo mediante el uso de redes de memoria a corto plazo (LSTM). Una de las ventajas de usar un modelo LSTM, es que se puede incorporar información adicional, así, se omite la necesidad de combinar modelos de predicción con heurísticas para, por ejemplo, considerar feriados nacionales o fines de semana. En este ámbito [17], investiga la aplicabilidad de los modelos LSTM para la predicción de la demanda de agua, y las compara con métodos utilizados actualmente por los proveedores de agua. Los resultados obtenidos muestran que el desempeño de los modelos LSTM es igual o mejor, en comparación con el tradicional Vector Autoregressive (VAR) y tiene la ventaja de necesitar poca información de entrenamiento para lograr buenos resultados. También se investigó el uso de redes preentrenadas, sin embargo, se obtienen peores resultados en comparación con uno entrenado desde cero. Además de evaluar los modelos de predicción en sí mismos, se realizó una simulación de un año de duración, en donde se comparó el modelo LSTM con un modelo que ya había demostrado su validez. Los resultados refuerzan la confianza en el modelo LSTM, que

se comportó de forma muy similar o incluso mejor que la referencia en todos los indicadores de rendimiento. Cabe destacar el excelente desempeño del modelo LSTM sin preentrenamiento, ya que permite a pequeños proveedores de agua que no cuentan con capacidad para almacenar grandes cantidades de datos históricos, a implementar soluciones factibles que permitan mejorar su operación.

En [18], los autores desean probar distintas técnicas para modelar la predicción de consumo de agua a corto plazo, utilizando intervalos de 1, 12 y 24 horas. En particular, se analiza el performance de técnicas basadas en Redes Neuronales Artificiales y Profundas (ANN y DNN, respectivamente), Extreme Learning Machines, Least Square Support Vector Machine (LSSVM), Procesos Gaussianos (GP), RF y regresión lineal múltiple (MLR) según un criterio de evaluación común, utilizando un dataset público de redes de sensores que miden el consumo de agua y temperatura en varios sectores de la Unión Europea. Los autores concluyen que para cualquier intervalo de tiempo, el modelo propuesto, basado ANN, provee el mejor desempeño, pero el tiempo de cómputo que involucra el entrenamiento de éstas puede llegar a ser perjudiciales para sistemas que necesitan obtener predicciones en tiempo real, por lo que sugieren como opción LSSVM que es un modelo más liviano, pero su performance es levemente peor que las obtenidas por ANN.

A continuación, en el Cuadro 3.1, se presenta un resumen las investigaciones estudiadas. Se debe considerar que para el método de entrenamiento se utilizará la abreviación a/b/c que indica la proporción porcentual del dataset que se ocupará para entrenamiento, desarrollo y pruebas respectivamente. En el caso de que se consideren estas dos últimas como un solo subset de pruebas, se ocupa la abreviación a/b.

Cuadro 3.1: Resumen de investigaciones de modelos de predicción.

| **Ref.** | **Modelos estudiados** | **Modelo propuesto** | **Características de los datos** | **Método de entrenamiento** |
|---|---|---|---|---|
| [14] | MLR, FFNN, RNN | Mútiples RNN (una por cada mes) | Consumo diario de agua en millones de galones en intervalos de 1 día, clima, temperatura máxima/minima, festivos e ingreso familiar promedio anual en el periodo 1997-2016. | Entrenamiento offline entre 1997-2015 para modelos no-NN. Para modelos basados en NN, se entrena offline desde 1997-2010 y online desde 2010-2015. Para todos los modelos, se ocupa los valores del año 2016 para efectos de pruebas. |
| [15] | ANN, ARIMA, DSHW, KNN | QMMP+ | Caudal $[m^3/\text{s}]$ de un año en intervalos de 1 [h] en estanques seleccionados | Offline, 70/30 |
| [16] | ANN, RF, KNN, SVR, ARIMA | Arquitecturas de ANN con distintos intervalos de muestreo | Caudal $[m^3/\text{s}]$ de 2 estanques, en periodos de 288 días y 352 días respectivamente, en intervalos de 1 [h]. | Offline, 96.875/3.125 |
| [17] | LSTM, VAR | Arquitecturas de LSTM | Caudal $[m^3/\text{s}]$ de 3 años, en intervalos de 1 [h]. | Offline, 33.3/33.3/33.3 |
| [18] | ANN, DNN, LSSVM, MR, GP | Arquitecturas de ANN | Lectura de temperaturas y consumo de agua entre los años 2016-2018, en intervalos de 1 [h]. | No especificado |
| Este trabajo | ARIMA, GP, MLR, Prophet | Determinado en tiempo de evaluación de performance | Volumen en $[m^3]$ de algunos estanques de la red de Esval en el periodo 2010-2019, en intervalos de 5 [min] | Offline, 70/30 |

En base a la literatura presentada, se puede observar que no existe un modelo de predicción único que sea el mejor bajo cualquier condición. El desempeño de éstos dependerá de muchos factores, en donde se destaca la cantidad y calidad de los datos, además de la complejidad de los distintos modelos ocupados.

Para resolver el problema de predicción asociado al desafío, es necesario considerar que se requiere una predicción a corto plazo del caudal de los estanques más estratégicos de la red y el tiempo tanto de respuesta como de entrenamiento deben ser cortos, por lo tanto, se utilizarán los modelos de SARIMA, GP, MLR y Prophet para evaluar cuáles de éstos tiene mejor desempeño utilizando el dataset dispuesto por la empresa.

## 3.4. Empresas del rubro

Debido a la existencia de un gran número de empresas que se dedican a crear software de optimización para diferentes tipos de industrias, no se puede definir una oferta nacional de manera cuantitativa, ya que es un mercado que está en constante crecimiento. A continuación se detallan 4 empresas dedicadas al desarrollo de software para la optimización de procesos.

### 3.4.1. SIMULART

Es un proveedor [19] de servicios profesionales y tecnologías para el proceso de toma de decisiones, basadas en prototipos virtuales y simulación computacional. Estas soluciones permiten mejorar el desempeño de sistemas productivos de materias primas, bienes y servicios en general. Provee a los tomadores de decisiones la oportunidad de poner virtualmente en funcionamiento el diseño de un nuevo proceso o servicio, para conocer anticipadamente cómo se desempeñará, corrigiendo los diseños si es necesario. También permite probar nuevas ideas de mejoramiento de un sistema existente, antes de invertir tiempo y recursos necesarios para construir o alterar el sistema actual. En general la empresa se dedica a la experimentación de soluciones a través de la simulación, con el fin de comprobar si cambios en el sistema actual suponen una mejora considerable.

### 3.4.2. NOTUS SPA

Esta empresa un spin off de DICTUC S.A [20] [21], el brazo tecnológico de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile. Notus ofrece servicios que generan un cambio tecnológico en la gestión de las empresas mediante la implementación de soluciones basadas en modelos de simulación y optimización. Se

especializa en desarrollar soluciones para optimizar y automatizar decisiones operativas, ha desarrollado soluciones para variadas industrias como, minería, salud, transporte y logística, y atención al cliente.

### 3.4.3. DataQu

Es una empresa [22] que entrega constantemente productos innovadores en el área de predicción y análisis, sin importar el tipo de industria o tamaño. Han desarrollado soluciones para la industria salmonera, seguridad, atención al cliente, logística y mercado financiero. Además de analizar datos, también diponibiliza la información relevante a través de aplicaciones de escritorio o en la nube.

### 3.4.4. TaKaDu

Es un líder [23] global en soluciones de Gestión Central de Eventos para empresas de agua. Brinda un servicio basado en la nube que permite a las empresas detectar, analizar y gestionar eventos e incidentes en la red, como fugas, roturas de matríz, activos defectuosos, problemas de telemetría y datos, fallas operativas, calidad del agua y más. El servicio actúa como la capa de administración central para todos los eventos de red, detectados por su propio motor de análisis de datos y otros sistemas de alerta externos. También permite la integración con otros sistemas informáticos, por ejemplo, GIS, órdenes de trabajo, CRM, call center y gestión de activos.

Cuadro 3.2: Servicios y funciones empresas del rubro.

| | **SIMULART** | **NOTUS SPA** | **DataQu** | **TaKaDu** |
|---|---|---|---|---|
| Simula operación del software | ✓ | ✓ | | |
| Optimizar decisiones operativas | ✓ | ✓ | ✓ | ✓ |
| Servicio en la nube | | | ✓ | ✓ |
| Predicción en línea | | | ✓ | ✓ |
| Gestión de eventos | | | | ✓ |

# Capítulo 4

# Desarrollo de la solución

En la creación del sistema que permita solucionar el problema planteado, es necesario identificar los componentes y flujos para la integración de estos. A modo general, se identifican dos flujos importantes: el de disponibilzación de los modelos y el de aplicación.

El flujo de disponibilzación de los modelos se muestra en la Figura 4.1. Primeramente se lleva a cabo el proceso de filtrado de los datos según criterios de la red, que involucra la selección de los datos relacionados a estanques de relevancia definidos por la empresa, además de descartar estanques que tienen datos defectuosos. Luego, se preprocesan los datos para poder ocuparlos como entrada a los modelos, involucrando suavizados a la serie de tiempo para manejar los outliers y adición de información extra como por ejemplo días feriados. Luego, se entrenan varios modelos definidos y se elije el mejor según criterios de error para cada modelo. Finalmente, se guardan los modelos en una base de datos que posteriormente la aplicación utilizará para disponibilizar las predicciones.

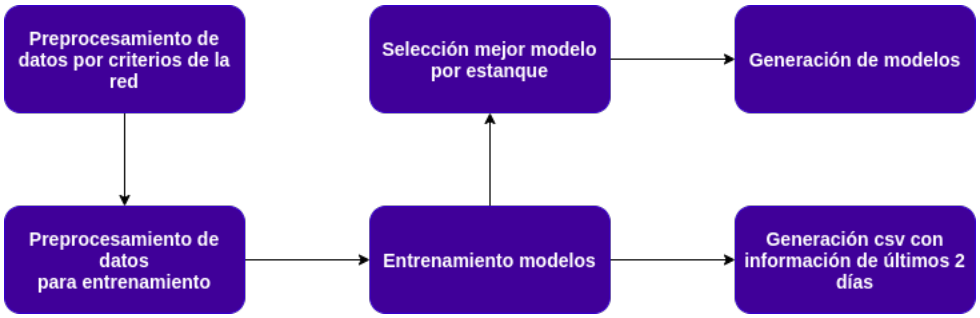



Figura 4.1: Flujo para obtener los modelos de predicción. Fuente: Elaboración propia.

## 4.1. Red de distribución

Dada la gran extensión de la red de distribución desplegada en el Gran Valparaíso, es que se decide hacer una simplificación en donde se consideren solo los estanques más relevantes de la red. La simplificación de la red se realizó en conjunto con miembros de Esval, en la Figura 4.2 se detallan las fuentes productivas y estanques de la red que se usarán para el desarrollo de la solución.

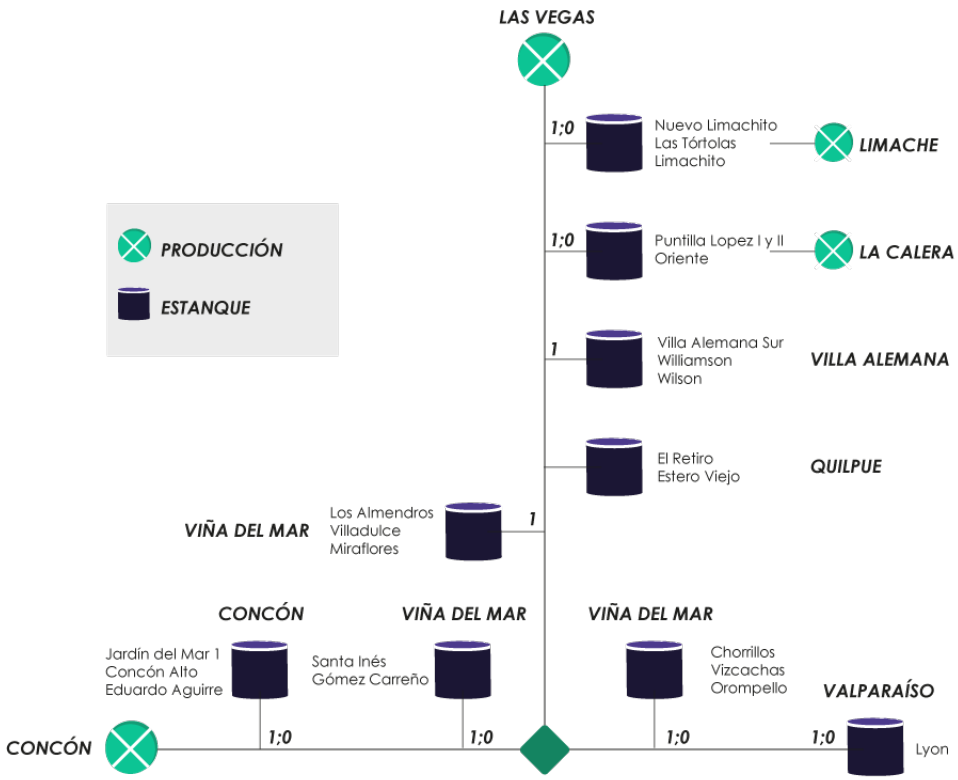



Figura 4.2: Red de distribución simplificada. Fuente: Elaboración propia.

Como se mencionó en 2, la Unidad de Telemetría de Esval controla la distribución de agua potable las 24 horas del día, esta también posee información histórica del volumen de agua de cada estanque de la red. A continuación en el Cuadro 4.1 se detalla la información de estanques solicitada a la Unidad de Telemetría, en donde el volumen actual de cada estanque está asociado a una etiqueta única que representa el sensor que lleva a cabo la medición. Toda la información de los sensores es gestionada con el software SCADA iFIX [24].

Cuadro 4.1: Resumen estanques y etiquetas.

| Localidad | Estanque | Etiqueta |
|---|---|---|
| Limache | Nuevo Limachito | SCADA001.EVAP_LIMACH_LVINS_MDC06.F_CV |
| Limache | Las Tórtolas | SCADA001.EVAP_LASTOR_LVINS_MDC03.F_CV |
| Limache | Limachito | SCADA001.EVAP_LIMACH_LVINS_MDC02.F_CV |
| Limache | Limachito | SCADA001.EVAP_LIMACH_LVINS_MDC03.F_CV |
| La Calera | Puntilla Lopez I y II | SCADA001.EVAP_CENTEN_LVINS_MDC02.F_CV |
| La Calera | Oriente | SCADA001.EVAP_ORIENT_LVINS_MDC02.F_CV |
| Villa Alemana | Villa Alemana Sur | SCADA001.EVAP_VALEM2_LVINS_MDC02.F_CV |
| Villa Alemana | Villa Alemana Sur | SCADA001.EVAP_VALEM2_LVINS_MDC03.F_CV |
| Villa Alemana | Williamson | SCADA001.EVAP_WILLIA_LVINS_MDC01.F_CV |
| Villa Alemana | Wilson | SCADA001.EVAP_WILSO1_LVINS_MDC02.F_CV |
| Quilpué | Estero Viejo | SCADA001.EVAP_EVIEJ1_LVINS_MDC03.F_CV |
| Quilpué | Estero Viejo | SCADA001.EVAP_EVIEJ1_LVINS_MDC04.F_CV |
| Quilpué | El Retiro | SCADA001.EVAP_RETIRO_LVINS_MDC01.F_CV |
| Viña del Mar 1 | Los Almendros | SCADA001.EVAP_ALMEND_LVINS_MDC01.F_CV |
| Viña del Mar 1 | Los Almendros | SCADA001.EVAP_ALMEND_LVINS_MDC02.F_CV |
| Viña del Mar 1 | Villa Dulce | SCADA001.EVAP_VDULCE_LVINS_MDC01.F_CV |
| Viña del Mar 1 | Miraflores | SCADA001.EVAP_EMIRAF_LVINS_MDC02.F_CV |
| Viña del Mar 1 | Miraflores | SCADA001.EVAP_EMIRAF_LVINS_MDC03.F_CV |
| Viña del Mar 2 | Santa Inés | SCADA001.EVAP_STAINE2_LVINS_MDC10.F_CV |
| Viña del Mar 2 | Santa Inés | SCADA001.EVAP_STAINE2_LVINS_MDC12.F_CV |
| Viña del Mar 2 | Santa Inés | SCADA001.EVAP_STAINE2_LVINS_MDC07.F_CV |
| Viña del Mar 2 | Gómez Carreño | SCADA001.EVSDAP_GCARRE_LVINS_MDC01.F_CV |
| Concón | Jardín del Mar | SCADA001.EVAP_SJMAR2_LVINS_MDC01.F_CV |
| Concón | Concón Alto | SCADA001.EVEEAP_CCALTO_LVINS_MDC01.F_CV |
| Concón | Eduardo Aguirre | SCADA001.EVAP_EAGUIR_LVINS_MDC01.F_CV |
| Viña del Mar 3 | Chorrillos | SCADA001.EVAP_CHORRI_LVINS_MDC01.F_CV |
| Viña del Mar 3 | Vizcachas | SCADA001.EVSDAP_VIZCAC_LVINS_MDC01.F_CV |
| Viña del Mar 3 | Orompello | SCADA001.EVAP_OROMPE_LVINS_MDC01.F_CV |
| Valparaíso | Lyon | SCADA001.EVAP_ELLYON_LVINS_MDC04.F_CV |
| Valparaíso | Lyon | SCADA001.EVAP_ELLYON_LVINS_MDC01.F_CV |

## 4.2. Generación Dataset

### 4.2.1. Formato

El dataset proporcionado consta de 4 archivos .csv que contienen información del volumen de los estanques a una tasa de muestreo de 5 minutos desde 01/01/2014 a 30/06/2019. A continuación se detalla el formato en que viene la información, también se presenta un ejemplo del dataset en el Cuadro 4.2.

Cuadro 4.2: Ejemplo formato de datos.

| **TagName** | **TimeStamp** | **Value** | **Quality** |
|---|---|---|---|
| SCADA001.EVAP_CENTEN_LVINS_MDC02.F_CV | 2019-01-01 00:05:00 | 78.500350 | NaN |
| SCADA001.EVAP_CENTEN_LVINS_MDC02.F_CV | 2019-01-01 00:10:00 | 73.018135 | NaN |
| SCADA001.EVAP_CENTEN_LVINS_MDC02.F_CV | 2019-01-01 00:15:00 | 72.409740 | NaN |
| SCADA001.EVAP_CENTEN_LVINS_MDC02.F_CV | 2019-01-01 00:20:00 | 69.773229 | NaN |

- **TagName:** Corresponde a la etiqueta del sensor que realiza la medición.
- **TimeStamp:** Fecha y hora en que se realiza la medición.
- **Value:** Volumen de agua actual medido por el sensor en $[m^3]$.
- **Quality:** Calidad de la medición, actualmente no operativo.

### 4.2.2. Preprocesado

A continuación se detalla el proceso realizado para obtener el dataset final de entrenamiento/validación, proceso también conocido como limpieza de datos.

#### Filtrado por red

El primer paso contempla filtrar la información en base a la red propuesta, esto quiere decir, eliminar toda la información de etiquetas de estanques que no están presentes en la red, y también verificar que exista información de cada uno de los estanques que la componen. A partir de esto se genera un dataset que tiene información de 19 de los 21 estanques de la red, equivalente a 27 de 30 tags SCADA solicitados. No se encontró información de los estanques, Nuevo Limachito, Gómez Carreño y Eduardo Aguirre, por lo tanto no se considerarán en el desarrollo de la solución.

A continuación en la Figura 4.3 se evidencian las 27 series de tiempo obtenidas del dataset generado. La gráfica representa el volumen de agua del estanque medido por el sensor (representado por el tagname) en intervalos de 5 minutos desde el año 2014 a 2019.

Es importante destacar que varias de las gráficas evidencian un comportamiento anormal, como valores fijos en cero por largos periodos y picks en las mediciones.

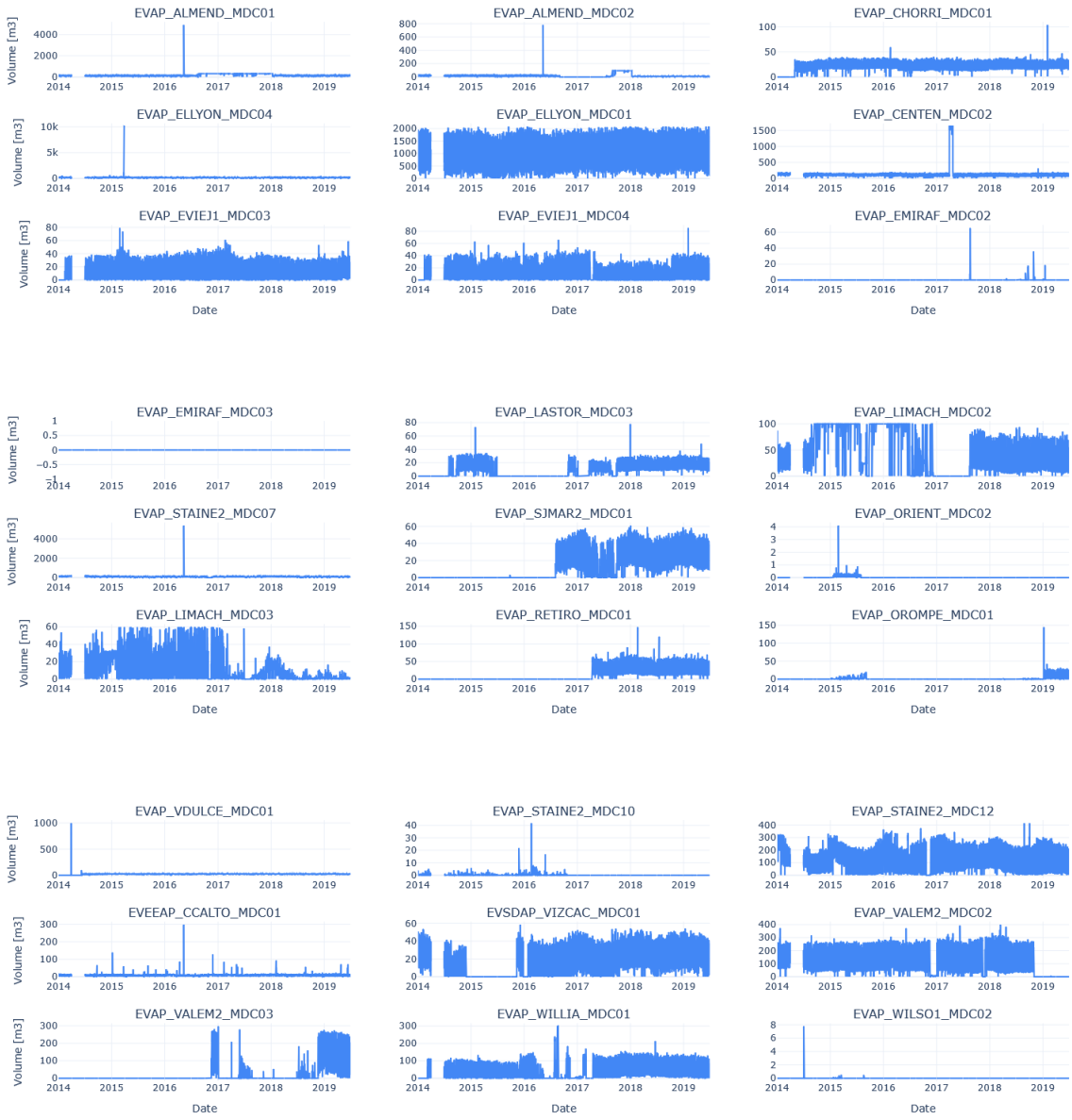



Figura 4.3: Series de tiempo periodo 2014 - 2019.

**Selección dataset entrenamiento**

Para el desarrollo de la solución se considera sólo la información del año 2019, ya que es el periodo en el que se encuentra información estable en la mayoría de los tags solicitados. A continuación en la Figura 4.4 se presentan las series de tiempo por

tagname en el período 2019. Mediante un análisis visual es fácil establecer que 5 de las 27 series de tiempo no poseen información en el periodo estudiado, por lo tanto estos no se considerarán al momento de agrupar la información para simular la red de distribución simplificada descrita anteriormente.

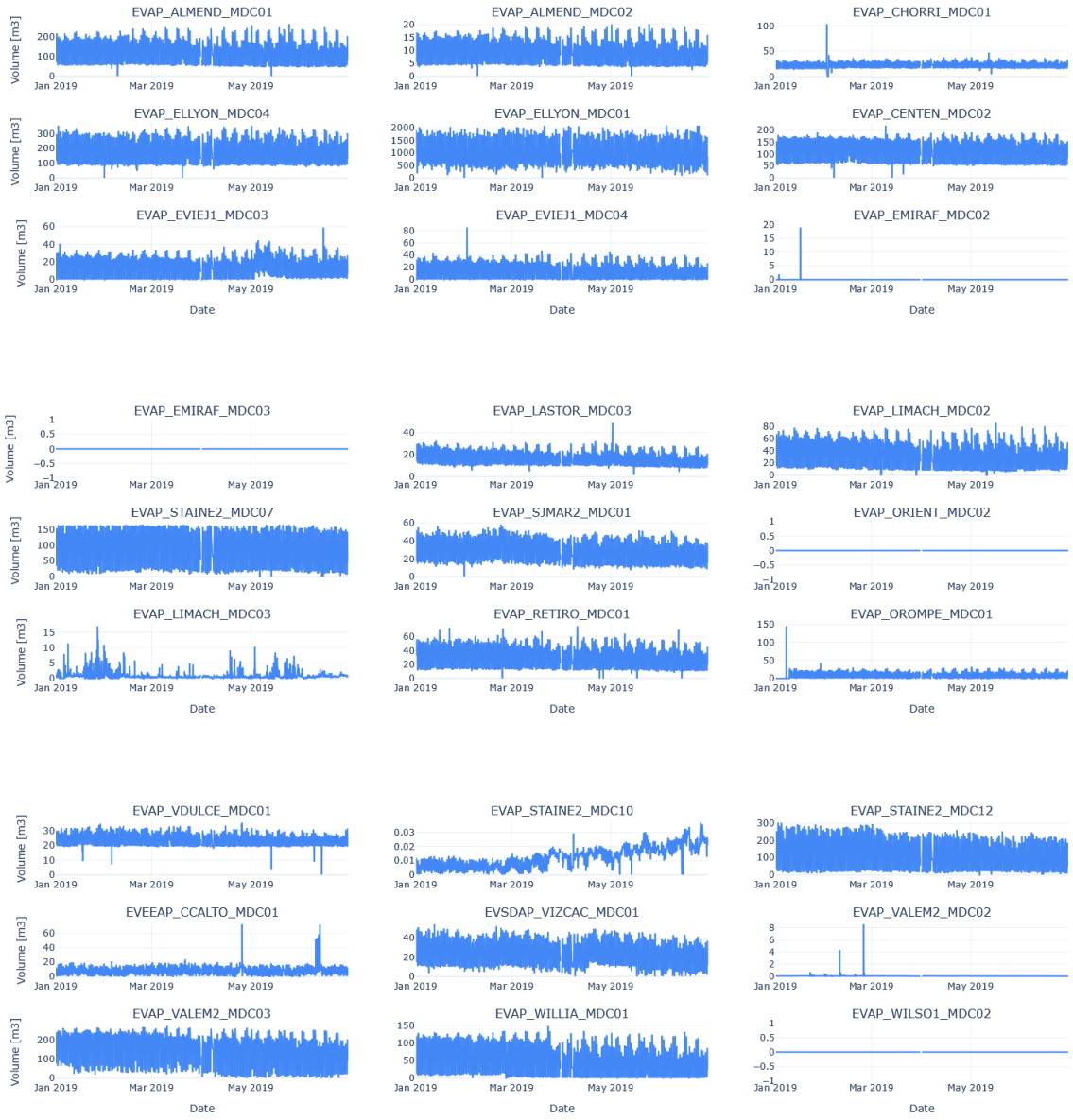



Figura 4.4: Series de tiempo periodo 2019.

A continuación en la Figura 4.5 se presentan las series de tiempo que representan el volumen de agua de los 9 estanques presentes en la red de distribución. Estas series de tiempo se generaron a partir de la suma de la información de los distintos tagname detallados en el Cuadro 4.1.

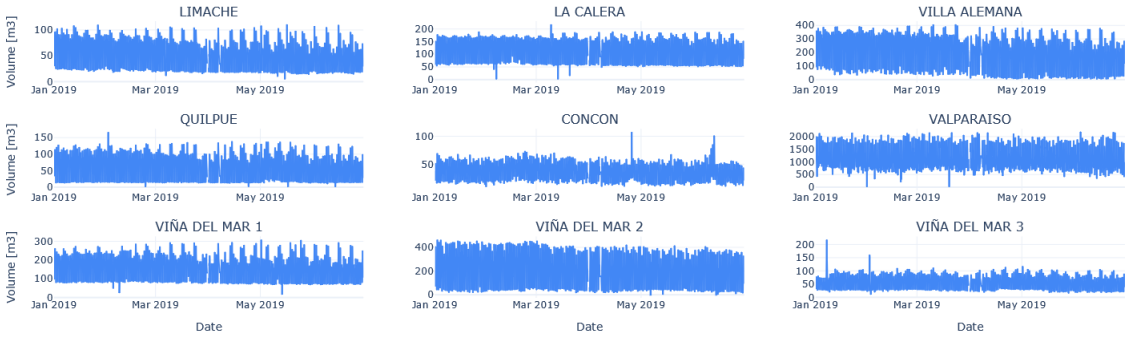



Figura 4.5: Series de tiempo referentes a los estanques de red simplificada.

### Corrección serie de tiempo

En la figura 4.5, se muestran las series de tiempo de los estanques de la red, en donde a simple vista se distinguen algunos outliers. Para el proceso de entrenamiento es indispensable utilizar información sin errores, es por esto que se realiza una última limpieza con el objetivo de remover estos outliers del dataset.

Para realizar el proceso de limpieza se decide usar un filtro de Hampel. Este método toma una ventana de tiempo de un ancho definido, y se recorre a lo largo de la serie de forma iterativa. Por cada iteración se calcula la mediana y la desviación estándar de la ventana usando la relación $\sigma = 1{,}4826 * MAD$, donde MAD es la desviación mediana absoluta definida por:

$$MAD = median(|X_i - median(X)|) \tag{4.1}$$

siendo $X = \{ X_i \} \, {}_{i=1}^{n}$ el set de datos.

Por cada punto de la ventana, si el valor calculado es mayor a $3\sigma$, dicho punto se detecta como outlier y se corrige con la mediana de la ventana.

En la limpieza realizada se consideró un ancho de 10 puntos y criterio de reemplazo de $3\sigma$. El dataset final con este criterio aplicado se presenta en la Figura 4.6.

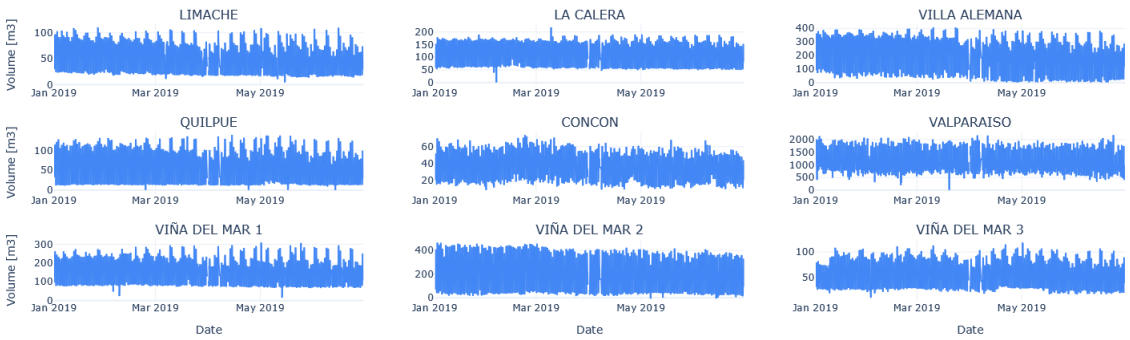



Figura 4.6: Series de tiempo con filtro de Hampel aplicado.

## 4.3. Predicción

En la presente sección se detallan los 4 modelos utilizados para realizar la predicción de la demanda a partir del dataset descrito anteriormente.

### 4.3.1. SARIMA

Es una extensión del modelo ARIMA, el cual modela series de tiempo con componentes estacionales. Como ARIMA por sí solo no soporta la predicción de series de tiempo con ciclos repititivos (componente estacional), se opta por usar SARIMA dado que la información con la que se trabaja tiene esta componente.

ARIMA se basa en la auto-regresión, el cual es un proceso de regresión de una variable en valores pasados de si misma donde las autocorrelaciones decaen gradualmente y estiman el grado en que el ruido blanco caracteriza una serie de datos. ARIMA se compone de tres parámetros (**p,d,q**), que modelan la tendencia de la serie, estos parámetros se detallan a continuación:

- **p:** Componente de auto regresión. Se utiliza un número de observaciones retrasadas de la serie para hacer las predicciones, se aplica un peso a cada uno de los términos pasados y los pesos pueden variar según lo reciente que sean. El valor p representa cuántos términos retardados serán usados en el modelo.

- **d:** Diferencia de tendencia. Si existe tendencia, entonces la serie de tiempo se considera no estacionaria. Este parámetro reduce la estacionalidad de la serie, a mayor valor, más reduce la estacionalidad. Los modelos ARIMA tienen un grado de diferenciación que elimina la estacionalidad.

- **q:** Media móvil. Este componente elimina el no determinismo o los movimientos aleatorios de una serie de tiempo. Especificamente el valor **q** representa las observaciones previas que se usan para calcular la observación actual.

Para representar los elementos estacionales de una serie de tiempo, SARIMA añade cuatro nuevos parámetros además de los anteriormente descritos, **(P,D,Q)m**. Los parámetros **(P,D,Q)** representan lo mismo que los parámetros **(p,d,q)** pero para estacionalidad envés de tendencia, y el parámetro **m** representa el número de pasos para un período estacional, por ejemplo un m de 12 se utiliza para datos que poseen una estacionalidad mensual.

En resumen, un modelo SARIMA es representado por 7 hiperparámetros,

$$(p, d, q)x(P, D, Q)m \tag{4.2}$$

donde los primeros 3 representan la tendencia de la serie, y los otros 4 representan la estacionalidad de la serie.

Existen infinitas configuraciones para definir un modelo, por lo que se opta realizar una búsqueda de hiperparámetros, tomando en cuenta un **m** fijo igual a 7, lo cual representa una estacionalidad semanal.

### 4.3.2. Prophet

Es una biblioteca desarrollada por Facebook con el fin de proveer una herramienta que permita analizar y predecir el comportamiento de series de tiempo fácilmente. La definición de Prophet por el mismo equipo desarrollador es la siguiente:

> *"Prophet es un método para pronosticar datos de series de tiempo basados en un modelo aditivo en el que las tendencias no lineales se ajustan a la estacionalidad anual, semanal y diaria, además de tomar en cuenta los efectos de los feriados."*

El modelo aditivo en cual se basa Prophet es el siguiente:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{4.3}$$

- g(t) modela la tendencia, que describe el aumento o la disminución a largo plazo de los datos.

- s(t) modela la estacionalidad con series de Fourier, que describen cómo los datos se ven afectados por factores estacionales como la época del año.

- h(t) modela los efectos de los días festivos o grandes eventos que afectan la estacionalidad de las series.

- $\epsilon_t$ representa el error.

Esta biblioteca se usará como caja negra en el desarrollo de la aplicación.

### 4.3.3. Regresión Lineal

Este modelo asume que los valores a predecir corresponde a una combinación lineal entre las características que representan la serie. Si se tiene un conjunto de $n$ observaciones $\{\mathbf{x_i}, y_i\}_{i=1}^{n}$ , donde $\mathbf{x_i} = (1, x_{i1}, ..., x_{ip}) \in \mathbb{R}^p$ corresponde a una fila $i$ del dataset de entrenamiento $\mathbf{X}$ con $p$ features, $y_i$ denota el valor de salida de la observación $i$, $\mathbf{w} = (w_0, ..., w_p) \in \mathbb{R}^p$ denota el peso de cada feature (para cualquier fila) y $\epsilon_i$ es un escalar de variables no observadas aleatorias (errores) que dan cuenta de la discrepancia entre el valor real y el predicho. Teniendo lo anterior en cuenta, el valor predicho $\hat{y_i}$ tiene la expresión:

$$\hat{y_i}(\mathbf{w}, \mathbf{x_i}) = w_0 + w_1 x_{i1} + ... + w_p x_{ip} + \epsilon_i = \mathbf{w}^T \mathbf{x_i} + \epsilon_i$$

El modelo se puede escribir en notación matricial, obteniendo la siguiente expresión:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + \epsilon$$

donde $\hat{\mathbf{y}} = (\hat{y_1}, ..., \hat{y_n}) \in \mathbb{R}^n$.

Para encontrar $\mathbf{w}$ tal que minimice el error de predicción se necesita definir una función de costo $J(\mathbf{w})$ que esta sujeta a los datos de entrenamiento $(\mathbf{X}, \mathbf{y})$, es decir,

$J(\mathbf{w}|\mathbf{X}, \mathbf{y})$ para luego minimizar ésta usando el método de mínimos cuadrados ordinarios.

La función de error típica corresponde a la del error cuadrático medio (ECM), definida por $J(\mathbf{w}|\mathbf{X}, \mathbf{y}) = ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2$. Es posible agregar términos a $J$ para penalizar la complejidad del modelo, con el propósito de reducir la complejidad de éste y en consecuencia, mejorar su capacidad de generalización y no caer en un sobreajuste.

Matemáticamente, la regularización en $J$ tiene la estructura $J = ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \alpha C(\mathbf{w})$, donde $C(\mathbf{w})$ es la medida de complejidad del modelo y $\alpha \geq 0$ es la importancia cuantitativa de la penalización.

Considerando que $||\mathbf{w}||_2$ denota la norma euclidiana de $\mathbf{w}$ en $\mathbb{R}^n$, mientras que $||\mathbf{w}||_1$ denota la norma L1 de $\mathbf{w}$, definida por $||\mathbf{w}||_1 = \sum_i |w_i|$, las regularizaciones típicas son las siguientes:

- **Ridge** Tiene una función de costo $J = ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \alpha||\mathbf{w}||_2^2$. Dicha regularización puede servir cuando se sospeche que varias características sean irrelevantes. Es decir, las características irrelevantes se le asigna un peso igual a 0, obteniendo un modelo que generalice mejor mediante la reducción de dimensión.

- **Lasso** Tiene una función de costo $J = \frac{1}{2n_{\text{samples}}}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \alpha||\mathbf{w}||_1$. Dicha regularización puede servir cuando se sospeche que varias características estan correlacionadas entre sí.

### 4.3.4. Procesos Gaussianos

Un GP es un proceso aleatorio donde cualquier punto $\mathbf{x} \in \mathbb{R}^d$ es asignado a una variable aleatoria $f(\mathbf{x})$ y donde la distribución conjunta de un número finito de variables $p(f(\mathbf{x}_1), ..., f(\mathbf{x}_N))$ es también Gaussiana, es decir:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \mathbf{K}) \tag{1}$$

donde $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \in \mathbb{R}^{nxd}$, $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))$, $\boldsymbol{\mu} = (m(\mathbf{x}_1), ..., m(\mathbf{x}_N))$ y

$K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. $m$ es la función de media y es común establecerla como $m(\mathbf{x}) = 0$ ya que los GP son lo suficientemente flexibles como para modelar la media arbitrariamente bien. $\kappa$ es una función de kernel o también llamada función de covarianza. Note que la matriz $\mathbf{K}$ debe estar definida semipositiva y ser simétrica. Así, un GP es una distribución sobre funciones cuya forma está definida por $\mathbf{K}$. Si los puntos $\mathbf{x}_i$ y $\mathbf{x}_j$ son considerados similares por la función de kernel, se espera que $f(\mathbf{x}_i)$ y $f(\mathbf{x}_j)$ sean similares.

Es posible estimar el GP posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ a partir del GP anterior $p(\mathbf{f}|\mathbf{X})$ después de haber tenido observaciones $(\mathbf{X}, \mathbf{y})$, por lo que el GP posterior se puede ocupar para obtener predicciones $\mathbf{f}_*$ dada una nueva entrada $\mathbf{X}_* \in \mathbb{R}^{mxd}$, donde $m$ es la cantidad de nuevos datos:

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) &= \int p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y})\, d\mathbf{f} \\ &= \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \end{aligned} \tag{2}$$

La ecuación $(2)$ corresponde a la distribución de la predicción que también es Gaussiana con media $\boldsymbol{\mu}_*$ y covarianza $\boldsymbol{\Sigma}_*$. Por la definición de GP, la distribución conjunta entre los datos observados $\mathbf{y}$ y las predicciones $\mathbf{f}_*$ es

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right) \tag{3}$$

Con $n$ datos de entrenamiento y $m$ nuevos datos de entrada, $\mathbf{K}_y = \kappa(\mathbf{X}, \mathbf{X}) + \sigma_y^2\mathbf{I} = \mathbf{K} + \sigma_y^2\mathbf{I} \in \mathbb{R}^{nxn}$, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*) \in \mathbb{R}^{nxm}$ y $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*) \in \mathbb{R}^{dxd}$. $\sigma_y^2$ es el término de ruido en la diagonal de $\mathbf{K_y}$. Dicho término se define en cero si los datos de entrenamiento están libre de ruido y en caso contrario se define con un valor mayor a cero. La media y la función de kernel para la distribución de la predicción posterior $\boldsymbol{\mu}_*$ y $\boldsymbol{\Sigma}_*$ pueden ser calculadas como muestra la ecuación $(4)$ y $(5)$.

$$\boldsymbol{\mu_*} = \mathbf{K}_*^T\mathbf{K}_y^{-1}\mathbf{y} \tag{4}$$

$$\boldsymbol{\Sigma_*} = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}_y^{-1}\mathbf{K}_* \tag{5}$$

Note que los hiperparámetros a definir para este modelo es la función de kernel $k(x_i, x_j)$ y el ruido $\sigma_y^2$. Para efectos de pruebas, se ocupará de kernel la Función de base radial definida por:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)) \tag{6}$$

Esta función agrega dos hiperparámetros: el parámetro $l$ controla la suavidad de la función mientras que $\sigma_f$ controla la variación vertical. La elección de $l$ y $\sigma_f$ no es problema, ya que las herramientas a ocupar como scikit-learn se dedican a encontrar el valor óptimo en función de los datos.

### 4.3.5. Guardado de modelos

Una vez finalizado el entrenamiento de todos los modelos, estos se prueban con el dataset de testeo y se obtienen los desempeños de cada uno para luego compararlos y guardar el con mayor rendimiento. Este proceso se realiza para cada uno de los estanques. Además del modelo entrenado, también se guarda un archivo .csv, el cual contiene la información de los últimos 2 días del dataset de testeo. Tanto los modelos entrenados como los archivos .csv son utilizados por la aplicación al momento de realizar las predicciones.

## 4.4. Elecciones de modelo y tecnologías

En esta sección se define la arquitectura de la aplicación que permite el acceso a la predicción diaria de consumo de agua por estanque mediante una interfaz de usuario. Para ello, se definen conceptos y tecnologías a utilizar, junto con su correspondiente razonamiento del porqué han sido elegidas. Se realiza un análisis comparativo con las opciones disponibles en el mercado por el lado de desarrollo frontend, backend y bases de datos para concluir cuáles se ajustan mejor a los requerimientos. Además, se aterrizan los requerimientos de la aplicación y se define los flujos de la

aplicación, considerando la interacción cliente-servidor.

**Tecnologías**

La aplicación debe proveer de predicciones por día por cada estanque mediante una interfaz de usuario intuitiva. Para ello, es necesario construir una lógica tal que permita el almacenamiento de los modelos en alguna base de datos, además de proveer una aplicación para poder acceder a las predicciones diarias. Para ello se utiliza una arquitectura cliente-servidor, o también llamada arquitectura de dos niveles que consiste en distribuir las tareas entre los proveedores de recursos o servicios y los consumidores de dichos recursos, los cuales son llamados **servidores** y **clientes** respectivamente. Esta división entre dos capas permite, entre otras ventajas, aislar la lógica de tratamiento de datos y el consumo de éstos.

Es necesario utilizar tecnologías que sean flexibles y escalables, ya que continuamente se deben ajustar a los requerimientos del cliente que pueden cambiar a través del tiempo. Para ello, se aísla cada componente de software según la lógica que debe implementar cada uno de éstos según una arquitectura modular. Para la aplicación a desarrollar se utilizarán dos componentes:

- **Backend:** En este componente se implementa la lógica de gestión de recursos y servicios de la aplicación. Es habitual el consumo de bases de datos y provisión de sistemas de autenticación.
- **Frontend:** En este componente se implementa la lógica de visualización de la aplicación. Utiliza los servicios provistos por el backend para mostrar información mediante una interfaz gráfica.

Es necesario considerar los siguientes términos que se utilizan a la hora de explorar las tecnologías a utilizar, los cuales corresponden a:

- **Lenguaje de programación:** Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes que

permiten controlar el comportamiento físico y lógico de una máquina. Dichos lenguajes entregan como resultado varios tipos de código de máquina para que ésta pueda interpretar las ordenes adecuadamente.

- **Framework:** Es una abstracción en la que el software, que proporciona una funcionalidad genérica, puede cambiarse selectivamente mediante código adicional escrito por el usuario, proporcionando así software específico de la aplicación. Proporciona una forma estándar de crear e implementar aplicaciones y es un entorno de software universal y reutilizable que proporciona una funcionalidad particular como parte de una plataforma de software más grande para facilitar el desarrollo de soluciones de software. Un framework puede incluir programas de soporte, compiladores, bibliotecas de códigos, conjuntos de herramientas que reúnen todos los componentes para permitir el desarrollo de un sistema.
- **Stack tecnológico:** Es una lista de todos los servicios tecnológicos utilizados para construir y ejecutar una sola aplicación.

### 4.4.1. Frontend

Para el análisis de tecnologías en este componente de software, se tendrá énfasis en las tecnologías más utilizadas en el área de desarrollo de aplicaciones en los últimos años y que permitan cierta escalabilidad. Según la encuesta anual realizada a desarrolladores en 2020 por Stack Overflow [25], las tecnologías más populares son **React** (35.9 %), **Angular** (25.1 %) y **Vue** (17.3 %).

Dichas tecnologías tienen aspectos en común: utilizan el lenguaje de programación Javascript, utilizan la programación reactiva que permite la reacción de la aplicación cada vez que haya un cambio en el flujo de datos. Además, éstas permiten la codificación de componentes bajo una arquitectura modular, la cual permite implementar la lógica de los distintos componentes de forma aislada y que ayuda a organizar y estructurar el código de la aplicación de mejor forma.

A continuación se realizará una comparación al detalle entre éstas, considerando sus ventajas y desventajas.

**React**

React [26] es una biblioteca escrita en Javascript de código abierto creada por Facebook, diseñada para crear interfaces de usuario mediante componentes, en donde éstos pueden ser cada una de las piezas que forman la interfaz o bien la interfaz completa. Cada componente contiene la lógica de tratamiento de los datos (si es que lo necesita) y de visualización. Es posible reutilizar los componentes en otros componentes, logrando así la disminución del tiempo de desarrollo.

**Ventajas**

- **Amplio ecosistema:** Cuenta con una amplia comunidad que dejan a disposición un gran número de bibliotecas externas con una muy buena documentación para las distintas funcionalidades que se requieran desarrollar.
- **Composición de componentes:** Las aplicaciones se elaboran con la composición de varios componentes donde se encapsula el comportamiento, una vista y un estado de una variable, por lo que la complejidad que tome el proyecto no es una preocupación, dado que el comportamiento reside dentro de cada componente y el resultado final será un conjunto de componentes generando una facilidad de mantenimiento, depuración y escalabilidad de la aplicación.
- **Flujo de datos unidireccional:** El patrón de funcionamiento que posee React permite que los componentes superiores propaguen los datos a los que están en un orden inferior. Éstos trabajarán con dichos datos y cuando cambian su estado según un criterio definido o eventos se notifica a los de orden superior para gatillar actualizaciones en la aplicación.

**Desventajas**

- **No ionizado:** Los desarrolladores a veces tienen muchas opciones de desarrollo, generando cierta confusión a la hora de decidir cuál es la opción adecuada.
- **Renderización:** El proceso de generación de la interfaz puede tomar más tiempo del estimado si es que hay grandes volúmenes de datos involucrados, provocando que la aplicación pueda ver afectado su rendimiento.
- **Integración del JSX:** Considerando que JSX es una extensión en React que permite mezclar sintaxis HTML con Javascript, esta puede producir confusión durante el desarrollo en el caso de que el desarrollador no tenga claro las diferencias entre código Javascript y HTML.

**Angular**

El lenguaje de etiquetado HTML es excelente para declarar documentos estáticos, pero falla cuando se intenta utilizar en vistas dinámicas, elemento fundamental en aplicaciones web. Angular [27] permite ampliar el vocabulario HTML para su aplicación. El entorno resultante es más expresivo, legible y rápido de desarrollar.

**Ventajas**

- **Desarrollado por Google:** Al tener un respaldo de parte de Google se puede estar seguro de que el framework es confiable y eficiente.
- **Basado en patrón MVC:** Utiliza el patrón de diseño MVC, la cual es muy útil cuando se desea programar una interacción con la página web y los datos proporcionados en capas distintas.
- **Enlaces de datos bidireccionales:** Significa que se puede realizar cualquier cambio relacionado con los datos e inmediatamente se propagaría a las vistas correspondientes y cuando se realiza cualquier cambio en la vista, eso también ocurriría en el modelo subyacente. Tan pronto como cambien los datos de la aplicación, también habrá cambios correspondientes en la interfaz de usuario.

#### Desventajas

- **Bibliotecas para Angular son muy específicas:** La inclusión de herramientas externas a Angular no funcionan adecuadamente.
- **Curva de aprendizaje lenta:** La dificultad de aprender Angular es alta, en comparación a otros frameworks similares.
- **Soporte de JavaScript obligatorio:** Si está deshabilitado Javascript en el navegador, los usuarios asociados no podrán acceder a su aplicación web.

### Vue

Vue [28] es un framework progresivo, lo que quiere decir que puede ser usado tanto para tareas básicas, como para tareas más complejas. A diferencia de otros frameworks monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La biblioteca central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras bibliotecas o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas Single-Page Applications (SPA) cuando se utiliza junto con herramientas modernas y bibliotecas de apoyo.

#### Ventajas

- **Gran Escala:** Se pueden desarrollar plantillas reutilizables como una alternativa más simple. Cualquier código HTML válido es también una plantilla Vue válida, haciendo mucho más fácil la migración progresiva de las aplicaciones.
- **Documentación detallada:** Está bien estructurada y completa, cubriendo todos los temas posibles, describiendo al detalle la instalación hasta aspectos más detallados, como la reactividad y el escalado de la aplicación.
- **Documentación de iniciación:** Vue tiene una documentación muy circunstancial que puede ajustar la curva de aprendizaje para los desarrolladores y ahorrar

mucho tiempo para desarrollar una aplicación utilizando solo los conocimientos básicos de HTML y JavaScript.

- **Rendimiento:** Vue puede mantener su velocidad y flexibilidad con un menor peso, en comparación con otros frameworks y bibliotecas.

**Desventajas**

- **Exceso de flexibilidad:** Vue puede tener problemas al integrarse en grandes proyectos y todavía no hay experiencia con posibles soluciones.
- **Falta de recursos:** Aún tiene poca cuota de mercado comparado con Angular o React, lo que significa que los recursos disponibles de este framework aún están en su fase inicial.
- **Ecosistema reducido:** La comunidad que mantiene Vue, a pesar de que ha aumentado estos últimos años, aún no es tan grande como otros frameworks como React o Angular. Lo anterior también afecta a la cantidad de bibliotecas disponibles para Vue.

### 4.4.2. Backend

Al seleccionar las tecnologías a ocupar en backend, es necesario considerar una restricción importante: tanto la compilación y uso de los modelos están bien documentados en entornos donde se utilice Python. Dado lo anterior, solo se seleccionarán frameworks y bibliotecas que utilicen dicho lenguaje de programación. Según la encuesta anual realizada a desarrolladores en 2020 por Stack Overflow [25], las tecnologías más populares para backend en Python son **Flask** (13.6 %) y **Django** (13.2 %).

A continuación, se realizará una comparación entre cada una de ellas, considerando aspectos como la documentación disponible, curva de aprendizaje y escalabilidad.

**Flask**

Flask es un framework web minimalista escrito en Python. Se le llama minimalista por el hecho que no necesita herramienta o bibliotecas particulares para funcionar, cumpliendo el objetivo de mantener el núcleo simple pero extensible. No cuenta con una capa de abstracción de base de datos, validación de formularios ni ningún otro componente donde las bibliotecas de terceros preexistentes proporcionan funciones comunes. Sin embargo, Flask admite extensiones que pueden agregar funciones de la aplicación como si estuvieran implementadas en el propio framework.

**Documentación disponible**

Flask cuenta con una documentación muy completa, que incluye desde tutoriales para crear proyectos básicos hasta aspectos de mayor complejidad.

**Curva de aprendizaje**

La curva de aprendizaje en Flask depende de la envergadura del proyecto, ya que es fácil de utilizar en aplicaciones pequeñas pero para aplicaciones de mayor envergadura se hace más complicado ya que hay que implementar ciertas características como integración a bases de datos o tecnologías de autenticación. Sin embargo, Flask cuenta con una comunidad que provee de extensiones para agregar funcionalidades al framework, como por ejemplo, mapeadores relacionales de objetos, validación de formularios, manejo de carga, varias tecnologías de autenticación abierta.

**Escalabilidad**

Para muchas aplicaciones web, la complejidad del código es un problema menor que el escalamiento según el número de usuarios o entradas de datos esperadas. Flask por sí solo está limitado en términos de escala por el código de su aplicación, la base de datos que desea usar, la implementación de Python y el servidor web en el que se está ejecutando. Se tiene una limitante con respecto al escalado en Flask que son los proxies locales de contexto. Dependen del contexto que en Flask se define como

un hilo, proceso o greenlet. Si el servidor utiliza algún tipo de simultaneidad que no se basa en subprocesos o greenlets, Flask ya no podrá admitir estos proxies globales. Sin embargo, la mayoría de los servidores utilizan subprocesos, greenlets o procesos separados para lograr la simultaneidad, que son métodos bien soportados por la biblioteca WSGI que permite la comunicación entre el código de Python y el servidor web.

**Django**

Django es un framework de desarrollo web de código abierto escrito en Python que utiliza patrón de diseño conocido como MVC. Según su web oficial [29], "Django es un framework de alto nivel para la web en Python que fomenta un rápido desarrollo y un diseño limpio y pragmático. Construido por desarrolladores experimentados, se encarga de gran parte de las molestias del desarrollo Web, de modo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es libre y de código abierto".

**Documentación disponible**

Django ofrece una basta documentación, la cual incluye tutoriales y videos en diversos idiomas incluyendo inglés y español. Además, existe una comunidad muy activa manteniendo el framework actualizado constantemente.

**Curva de aprendizaje**

La principal ventaja de Django es que esta escrito en Python, un lenguaje que combina eficiencia con simplicidad, ya que éste cuenta con una sintaxis muy simple basada en legibilidad de código, es multiparadigma y es hoy por hoy uno de los lenguajes más usados para el ámbito de ciencias de datos. Al ser Python fácil de aprender, Django tiene un mayor alcance. La gran ventaja de Django es que incluye muchas aplicaciones integradas listas para usar, lo que ayuda a la adaptabilidad de las necesidades del negocio y al mismo tiempo lograr minimizar el tiempo de desarrollo en imple-

mentar funcionalidades que son conocidas y que tienden a ser tediosas. No obstante, lo anterior puede llegar a ser una desventaja si se da el caso que el desarrollador sea inexperto, ya que la utilización de muchas aplicaciones puede dificultar el manejo de éstas en un principio.

**Escalabilidad**

En este sentido Django tiene una gran ventaja, ya que en su documentación oficial enseña a como utilizar el framework tanto para aplicaciones pequeñas como grandes, demostrando su adaptabilidad y estabilidad. Además, como el framework incluye aplicaciones listas para utilizar, se pueden agregar funcionalidades rápidamente al aumentar la complejidad de los requerimientos del software final.

### 4.4.3. Bases de datos

Para manejar información tanto de los usuarios de la aplicación como de los estanques disponibles para consultar su predicción, es necesario crear una base de datos (BD) que sea capaz de mantener toda la información.

A la fecha hay dos paradigmas para el manejo de los datos, que corresponden a las bases de datos relacionales y no relacionales. Ambos paradigmas cuentan con sistemas que proveen el uso de éstas como por ejemplo MySQL, PostgreSQL y Mongo DB. En esta sección se centrará solamente en los paradigmas y no en los sistemas, ya que la elección de éstos dependen de la compatibilidad con las tecnologías a utilizar.

**Base de datos relacional**

Son una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas donde se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin necesidad de reorganizar las tablas de la base. La interfaz estándar de programa de usuario y aplicación a una base de datos relacional es el Lenguaje de Consultas Estructuradas (SQL). Los comandos SQL se utilizan tanto para

consultas interactivas como para obtener información de una base de datos relacional y la recopilación de datos para informes.

Las bases de datos relacionales se basan en la organización de la información en partes pequeñas que se integran mediante identificadores llamados llaves (o keys) que generan una relación o enlace entre los datos.

La principal desventaja de este tipo de base de datos, proviene de su rigidez en la estructura, debido a que el usuario debe definir previamente como estarán estructurados cada uno de los elementos, tanto en sus relaciones como en atributos.

**Base de datos no relacional**

Están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Son ampliamente reconocidas porque son fáciles de desarrollar, tanto en funcionalidad como en rendimiento a escala. Usan una variedad de modelos de datos, que incluyen documentos, gráficos, clave-valor, en-memoria y búsqueda.

Las bases de datos no relacionales (NoSQL) son las que, a diferencia de las relacionales, no tienen un identificador que sirva de relación entre un conjunto de datos y otros. La información se organiza normalmente mediante documentos y es muy útil cuando no tenemos un esquema exacto de lo que se va a almacenar.

Con relación a formatos, la información de una base de datos puede ser almacenada en tablas o documentos. Cuando los datos son organizados en un archivo de Excel, es en formato tabla, pero cuando simplemente son datos escritos como cartas, fórmulas o recetas, son datos en formato documento. Esto aplica para los dos tipos de bases de datos. Los datos almacenados en tablas pueden tener una relación entre una tabla y otra, pero en este paradigma la relación se tiene que configurar de forma manual.

A diferencia de las bases de datos relacionales, las no relacionales no tienen una estructura rígida para el manejo de datos, por lo que si se quisiera hacer cambios en las

estructura de los datos en etapas avanzadas de un desarrollo, no seria tan engorroso ya que difícilmente estos cambios afecten el trabajo previamente realizado.

### 4.4.4. Resumen de selección de tecnologías

**Frontend**

Para la elección del framework para el frontend se construyeron 3 gráficos en donde se comparan cada uno de los frameworks considerados bajo ciertos criterios.

En la Figura 4.7 se compara la curva de aprendizaje vs la experiencia de desarrollo, en donde tanto React como Vue tienen una curva de aprendizaje similar, sin embargo, React tiene mayores bibliotecas creadas por la comunidad que mejoran la experiencia de desarrollo. El peor para ambos criterios es Angular ya que su curva de aprendizaje es complicada, además de empeorar la experiencia de desarrollo dada la dificultad de agregar bibliotecas externas a Angular.

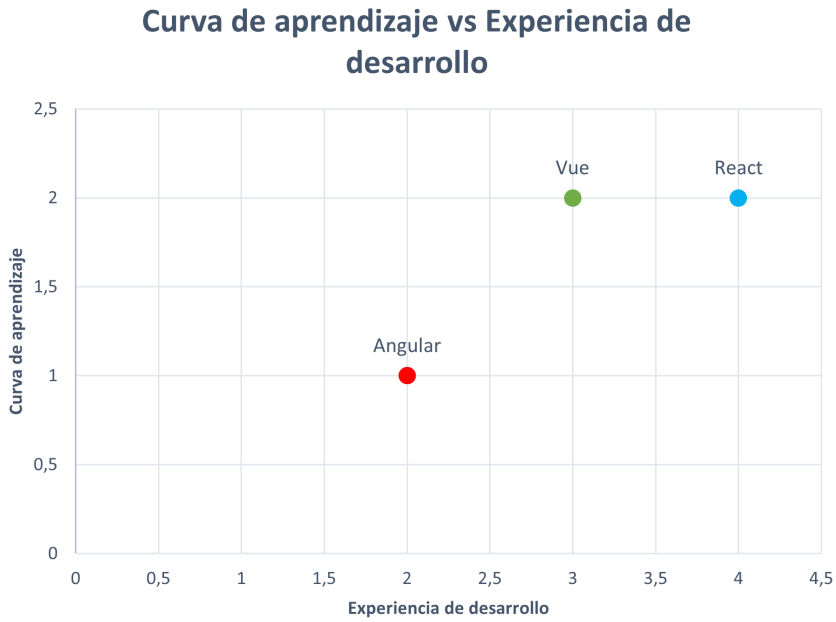



Figura 4.7: Curva de aprendizaje vs Experiencia de desarrollo.

En la Figura 4.8 se muestra la comparación entre claridad de código vs aplicación de gran escala. En términos de claridad de código React es superior ya que su arquitectura orientada a componentes y el uso de JSX permite una sintaxis limpia. Por otro lado, en términos de escalabilidad, Angular es mejor ya que está orientado a aplicaciones de gran escala, pero al tener tantas herramientas se da que la estructura

del código es compleja. React es buena opción para crear aplicaciones grandes, pero como dicho framework incluye menores herramientas integradas, hay mayores libertades en cómo estructurar la aplicación y ello puede ser problemático al momento que se necesite escalar la aplicación. Vue es el peor en dicha clasificación dado que, a pesar de que es un framework poderoso, el reuso de templates HTML puede ser más tedioso que el uso de JSX de React.

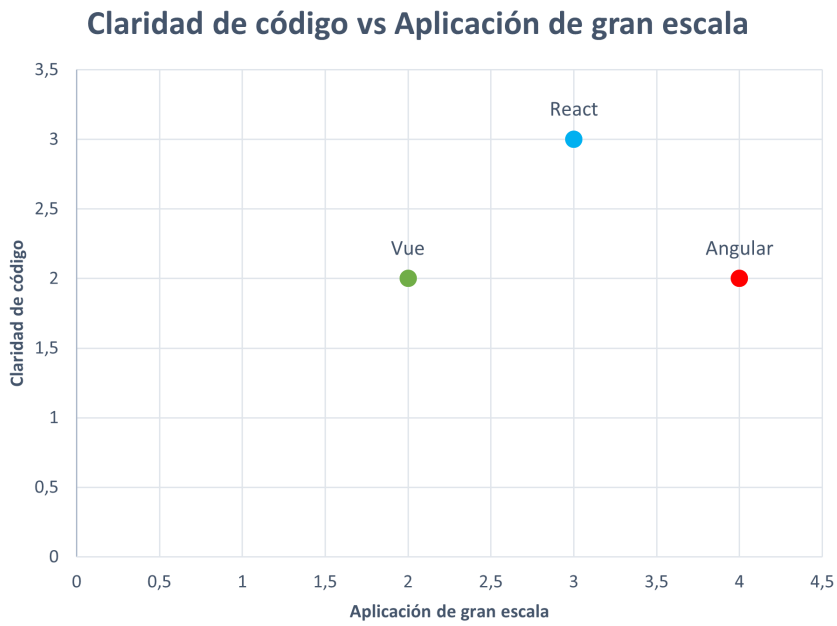



Figura 4.8: Claridad de código vs Aplicación de gran escala.

En la Figura 4.9 se compara Rendimiento vs Documentación. Por el lado del rendimiento, Angular es superior en rendimiento a ambos frameworks mientras que por el aspecto de la documentación, tanto React como Vue tiene una documentación muy completa, pero React predomina al tomar en cuenta que éste tiene mayor comunidad que constantemente mantiene y crea nuevas bibliotecas complementarias.

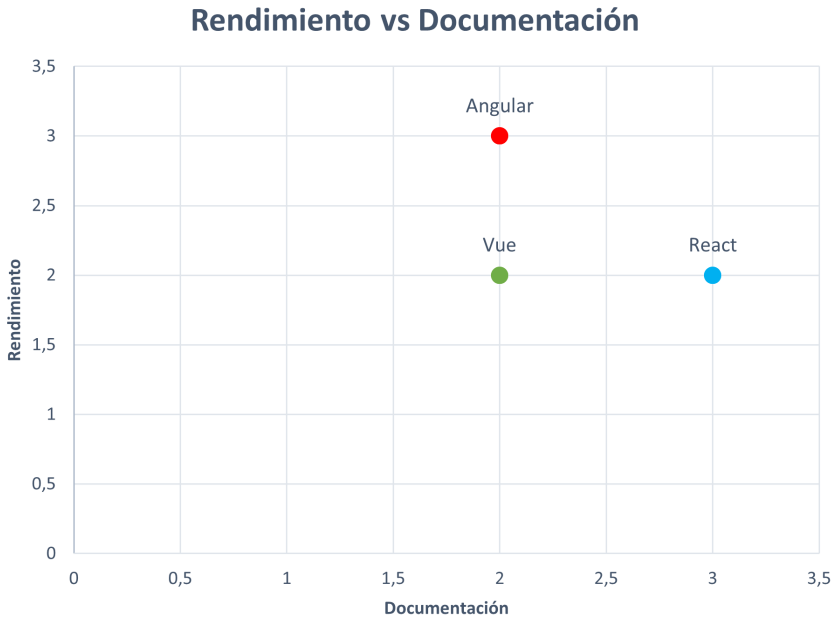



Figura 4.9: Rendimiento vs Documentación.

Considerando los aspectos estudiados, se obtiene el puntaje final por cada framework como se muestra en el Cuadro 4.3. Por lo tanto, se elige React como biblioteca para frontend ya que cuenta con una buena documentación, es fácil de utilizar y posee alta escalabilidad.

Cuadro 4.3: Puntaje final de cada framework.

| Frameworks de Frontend | Puntaje total |
|---|---|
| React | 17 |
| Angular | 14 |
| Vue | 13 |

**Backend**

Para la elección del framework de backend, aparte de los criterios de documentación disponible, curva de aprendizaje y escalabilidad, es necesario considerar también el conocimiento previo que se tiene de los frameworks. Cabe destacar que para el alcance que se tiene en términos de desarrollo, tanto Flask como Django satisfacen los requerimientos del sistema. A continuación se presentan los puntajes para dichos criterios:

**Conocimiento previo**

Se utiliza una escala del 1 al 10, en donde 1 significa un total desconocimiento del framework mientras que 10 representa un nivel alto de conocimiento.

**Documentación disponible**

Se representa con puntaje 1 una nula documentación oficial o por su comunidad, mientras que un puntaje 10 representa una documentación completa provista tanto por los desarrolladores como por la comunidad y segmentada según el nivel de conocimiento.

**Curva de aprendizaje**

En este criterio se considera tanto la arquitectura del framework como las bibliotecas complementarias desarrolladas por la comunidad. Se representa con puntaje 1 al framework que es difícil utilizar, requiere de un tiempo considerable y que no cuenta con bibliotecas que agilicen el desarrollo, mientras que el puntaje 10 significa que el framework es muy fácil de utilizar y hay bibliotecas disponibles que aportan fuertemente a la disminución del tiempo de desarrollo.

**Escalabilidad**

En este criterio se considera la capacidad de modularización del framework y casos de éxito en aplicaciones complejas. Se representa con puntaje 1 una modularización dificultosa y sin registros de aplicaciones grandes que utilizen el framework, mientras que el puntaje 10 significa que es fácil modularizar, además de tener variados registros de casos de éxito.

A modo de resumen, en el Cuadro 4.4 se muestra puntaje por los criterios mencio-

nados anteriormente, junto con el puntaje total para cada framework. Dado el puntaje total, se determina que se utilizará Django para el desarrollo del backend, el cual agilizará el desarrollo dada su amplia gama de bibliotecas que permiten utilizar características de alta complejidad de implementación de forma sencilla.

Cuadro 4.4: Resumen de puntajes para cada framework.

| | Django | Flask |
|---|---|---|
| Conocimiento Previo | 7 | 4 |
| Documentación disponible | 8 | 7 |
| Curva de aprendizaje | 6 | 7 |
| Escalabilidad | 8 | 6 |
| **Total** | 29 | 24 |

**Base de datos**

En base a que se decide usar Django para Backend, se debe considerar que dicho framework está preparado para ser usado junto con BD de tipo relacionales. Si bien existen bibliotecas que permiten la integración de Django con una BD no relacional, estas son limitadas y además son de pago. Por otro lado, dado el alcance del proyecto se requiere de una BD pequeña, por lo que el criterio a considerar radica netamente en la facilidad de integración entre ésta y Django. Por lo tanto, se decide utilizar una BD relacional para la aplicación. Junto a lo anterior, se necesita elegir un sistema de gestión de bases de datos (SGBD) que permita el almacenamiento, modificación y extracción de la información. Dada la poca complejidad de la aplicación, cualquier SGBD puede satisfacer los requerimientos, por lo tanto se utilizará PostgreSQL como SGBD dada su facilidad de integración con Django.

**Arquitectura final de la aplicación**

A modo de recapitulación del análisis ya explicado, en la Figura 4.10 se muestra la arquitectura de la aplicación utilizando los frameworks elegidos anteriormente. En particular, se utilizará por el lado del frontend React mientras que para el backend se utilizará Django, en donde se dispone una interfaz de programación de aplicaciones (API) que permite la comunicación frontend-backend mediante solicitudes

HTTP que retornan una respuesta (payload). La API ofrece el manejo de credenciales, comunicación con la base de datos PostgreSQL e información sobre estanques disponibles. Además, tiene acceso a directorios donde se almacenan los modelos de predicción por estanque junto con data de dos días anteriores para disponer de predicciones en el caso de los modelos basados en Machine Learning.

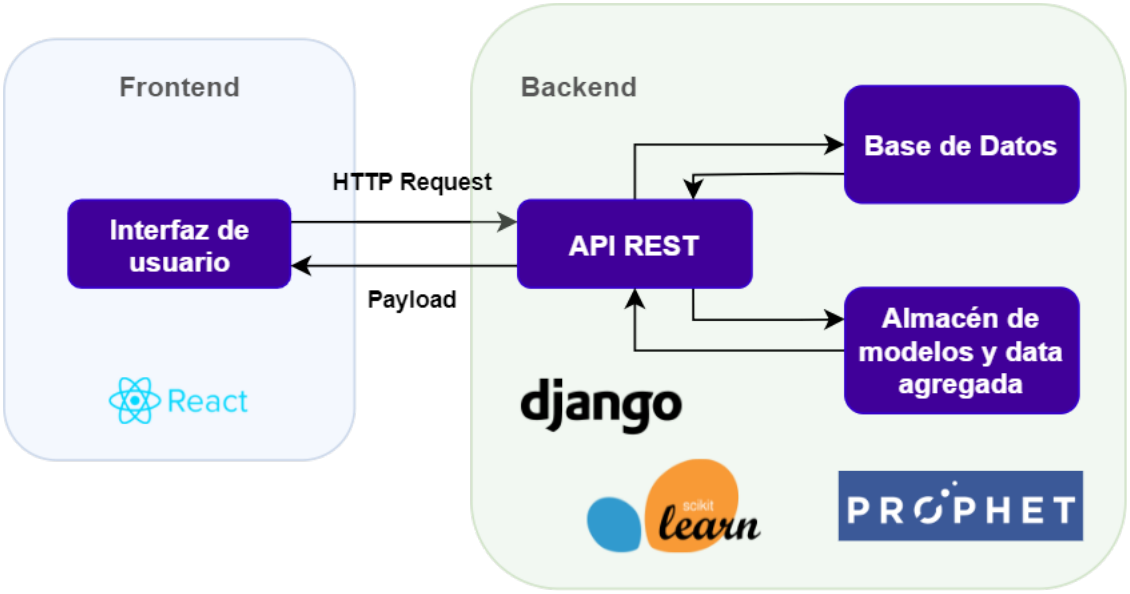



Figura 4.10: Arquitectura de la aplicación.

## 4.5. Capa de datos

La base de datos debe ser capaz de manejar datos de cuentas de usuario y estanques disponibles para obtener la predicción diaria. En la Figura 4.11 se muestra el diagrama entidad-relación que describe las relaciones entre las tablas y los campos que componen estas últimas.

Para el manejo de cuentas, Django incluye un sistema de manejo de cuentas de usuario lista para ocupar que nos permite autenticar y autorizar cuentas, además de crear, eliminar y editar éstas mediante una interfaz gráfica que el framework dispone. Para hacer posible lo anterior, el framework crea una serie de tablas con relaciones que están compuestas por todas las de la Figura 4.11, excepto `ponds_pond`.

Para el manejo de datos de estanques, Django permite crear distintas entidades (o tablas) y sus relaciones entre otras símiles mediante el ORM (Object Relation Mapper) dispuesto por el framework que encapsula el lenguaje SQL en funciones de Python lo cual facilita la construcción de la base de datos. En particular, para manejar los datos de estanques se crea una tabla `ponds_pond` que tiene los siguientes campos:

- `id`: Identificador numérico del estanque.
- `tagname`: Corresponde al nombre del estanque. Por ejemplo: VALPARAISO, QUILPUE.
- `model_platform`: Corresponde al nombre del framework que fue construido el modelo para dicho estanque. Tiene dos valores posibles: `sklearn` y `prophet`.
- `last_timestamp`: Marca de fecha y hora del último valor de volumen observado del estanque. Se usará la hora siguiente a dicha marca como el tiempo inicial de la ventana de predicción de 24 horas.

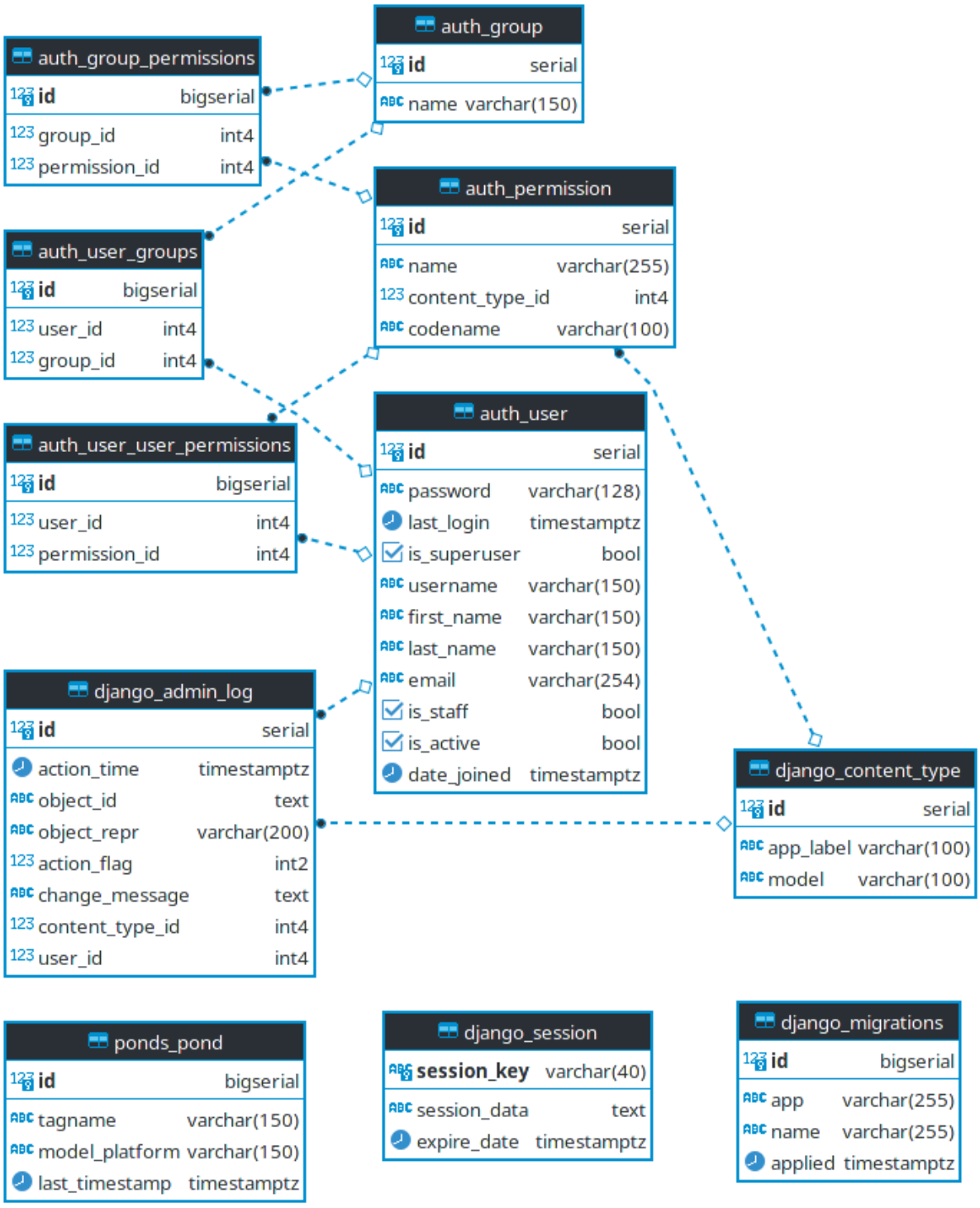



Figura 4.11: Diagrama Entidad-Relación de la aplicación.

## 4.6. Capa de negocios

Considerando la arquitectura mostrada en la Figura 4.10 que plasma los requerimientos del sistema, se construye una API en Django para habilitar la comunicación cliente-servidor. Dicha API ofrecerá un servicio de autenticación de cuentas, datos de estanques disponibles y la predicción de éstos, mediante el uso de la base de datos y el almacén de modelos y data agregada.

La API será desarrollada utlizando Django REST Framework que, entre otras características, permite la disponibilización de endpoints rápidamente. Además, se utilizará el plugin de este último framework llamado Simple JWT que permite la autenticación de usuarios mediante tokens.

Se muestra en el Cuadro 4.5 los endpoints a utilizar en la aplicación.

Cuadro 4.5: Tabla de endpoints de la aplicación

| **Descripción/Ruta** | **Método HTTP** | **Header** | **Body** |
|---|---|---|---|
| Iniciar sesión<br><br><host>/api/token/ | POST | Content-Type: application/json | {<br>username: string,<br>password: string<br>} |
| Obtener lista de estanques<br><br><host>/api/ponds | GET | Content-Type: application/json<br>Authorization: token | No aplica |
| Obtener predicción de un estanque<br><br><host>/api/pond/predict/:pondId | GET | Content-Type: application/json<br>Authorization: token | No aplica |

## 4.7. Capa de presentación

En esta sección se detalla la lógica implementada en el lado del cliente (frontend), considerando la comunicación con el backend mediante la API REST y el manejo de la información obtenida a través de la interfaz gráfica. Para generar HTTP Requests se utiliza la biblioteca `axios` que facilita la construcción y realización de las peticiones *request* para todo tipo de métodos HTTP, tales como *GET, POST, PUT, DELETE*. Dicha realización retorna una respuesta (*response*). Además, se le pueden agregar campos

a los *Headers* de las peticiones tal que permitan diversas funcionalidades, como por ejemplo autenticar a algún usuario en particular.

### 4.7.1. Mecanismo de Tokens de Acceso

Las API REST son vitales para el funcionamiento de aplicaciones y sistemas, pero es necesario protegerlas ante algún agente no autorizado ya que éstas se encuentran disponibles en la red y cualquiera podría utilizar los servicios provistos por las API. Hay distintos mecanismos para autenticar usuarios a endpoints protegidos, en donde se destaca la **autenticación por token**, el cual cuenta con fases de autorización y autenticación:

- **Autorización:** Consiste en obtener acceso a los recursos protegidos de la API en base a credenciales como por ejemplo usuario y contraseña, como se muestra en la Figura 4.12. En particular, se hace una petición HTTP desde el cliente a la API para solicitar un token de acceso que es una cadena de caracteres que es utilizada posteriormente para verificar la identidad del usuario. Si las credenciales son correctas la API envía el token, en el caso contrario se envía un mensaje de error.

- **Autenticación:** Consiste en solicitar recursos protegidos de la API utilizando el token obtenido en la fase de Autorización, como se muestra en la Figura 4.13. En dicha fase, el cliente envía junto a la petición HTTP el token a la API para que posteriormente ésta compruebe su validez, considerando tiempo de validez y a qué usuario le pertenece. De esta forma, se pueden utilizar endpoints que requieran autenticación. Si el token es válido se envían los datos solicitados al cliente, en el caso contrario se rechaza la solicitud.

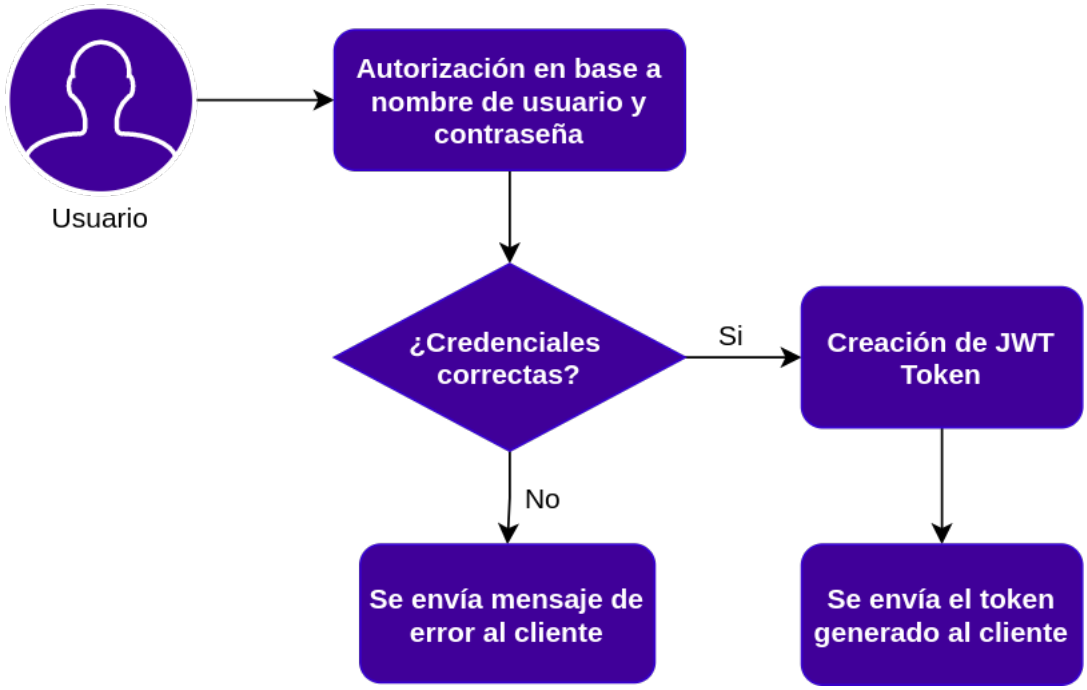



Figura 4.12: Diagrama de flujo de autorización de la aplicación.

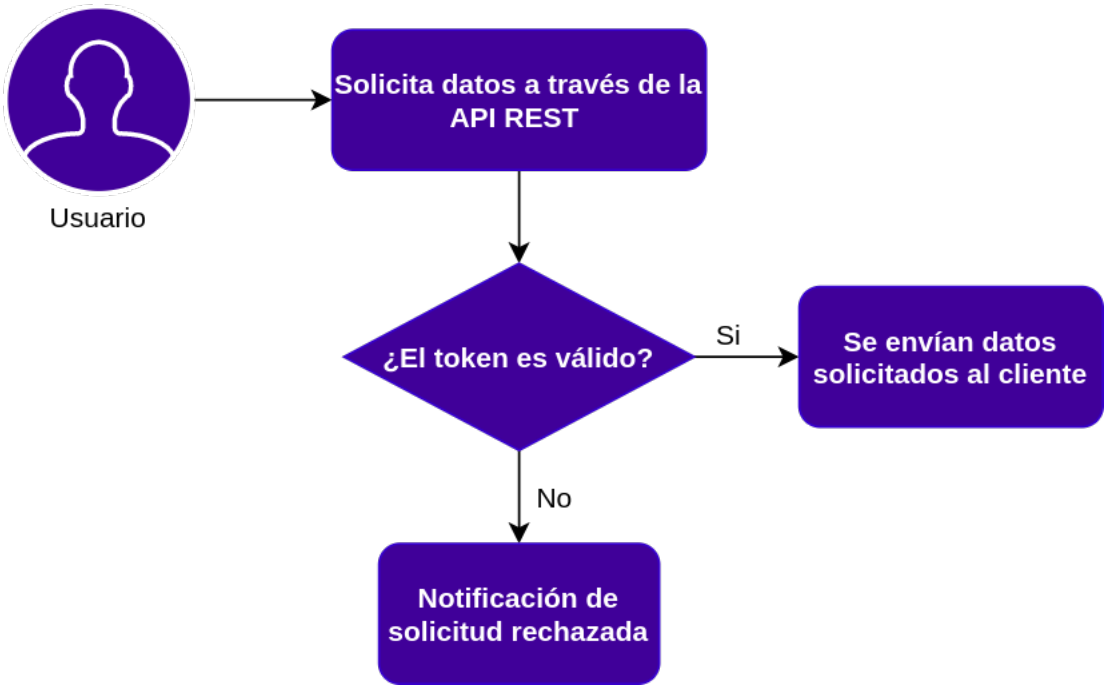



Figura 4.13: Diagrama de flujo de autenticación de la aplicación.

### 4.7.2. Vista Login

En la vista de login se muestra un formulario con los campos de nombre de usuario y contraseña, en donde éstos se encuentran mapeados a dos estados de la vista, los cuales corresponden a `username` y `password` respectivamente.

Al presionar el botón de *Ingresar* se ejecuta la lógica de autorización de la cuenta como se muestra en la Figura 4.14, la cual consiste en gatillar una petición HTTP con método *POST* a la URL `http://localhost:8000/api/token` con headers *'Content-Type': 'application/json'* para indicar que los datos de entrada están en formato JSON y en el *body* se agregan los estados `username` y `password`.

La respuesta a dicha solicitud pueden ser las siguientes:

- **Credenciales correctas:** La respuesta contiene un código HTTP 200, el cual significa que se ha logrado autorizar la cuenta. En la respuesta se incluye el token que será útil para autenticar las solicitudes posteriores, el cual será guardado en el navegador. Luego, se redirigirá al usuario al Dashboard de la aplicación.
- **Credenciales incorrectas:** En este caso la respuesta contiene un código HTTP 401, el cual significa que no se ha logrado autorizar la cuenta. Se muestra un mensaje de error en la vista en la espera del reingreso de credenciales.

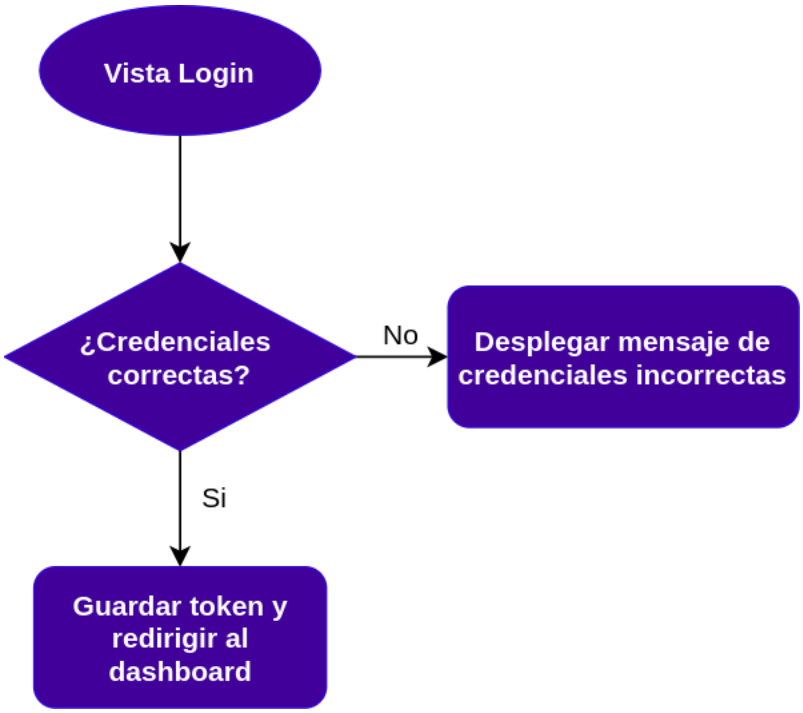



Figura 4.14: Diagrama de flujo de vista de login.

### 4.7.3. Vista Dashboard

En el dashboard se muestra la lista de estanques disponibles para obtener la predicción del volumen de éste en 24 horas, que es mostrada a través de un gráfico. Además, cuenta con un menu lateral donde el usuario puede cerrar sesión. La vista principalmente cuenta con dos estados, los cuales corresponden a

- `ponds`: Arreglo de estanques disponibles, por defecto se encuentra vacío. Cada elemento del arreglo tiene la estructura
  - `id`: Valor numérico que indica el identificador del estanque en la base de datos.
  - `tagname`: Valor de tipo string que indica el nombre del estanque.
- `predictionData`: Arreglo que contiene las predicciones del estanque seleccionado, por defecto se encuentra vacío. Cada elemento del arreglo tiene la estructura

- `timestamp`: Marca de tiempo de la hora de predicción.
- `value`: Valor numérico de volumen en el tiempo indicado en el atributo `timestamp`.

Al cargar la vista, se obtienen los estanques disponibles bajo la lógica mostrada en la Figura 4.15 gatillando una petición HTTP con método *GET* a la URL `http://localhost:8000/api/ponds` con headers *'Content-Type': 'application/json'* y *'Authorization': 'Bearer [TOKEN]'*, donde este último header se encarga de agregar el token obtenido en la fase de autorización para que posteriormente la API autentique al usuario. Mientras se realiza la petición, se muestra un mensaje que indica que la petición está en curso. Al finalizar la petición, se muestran los estanques disponibles.

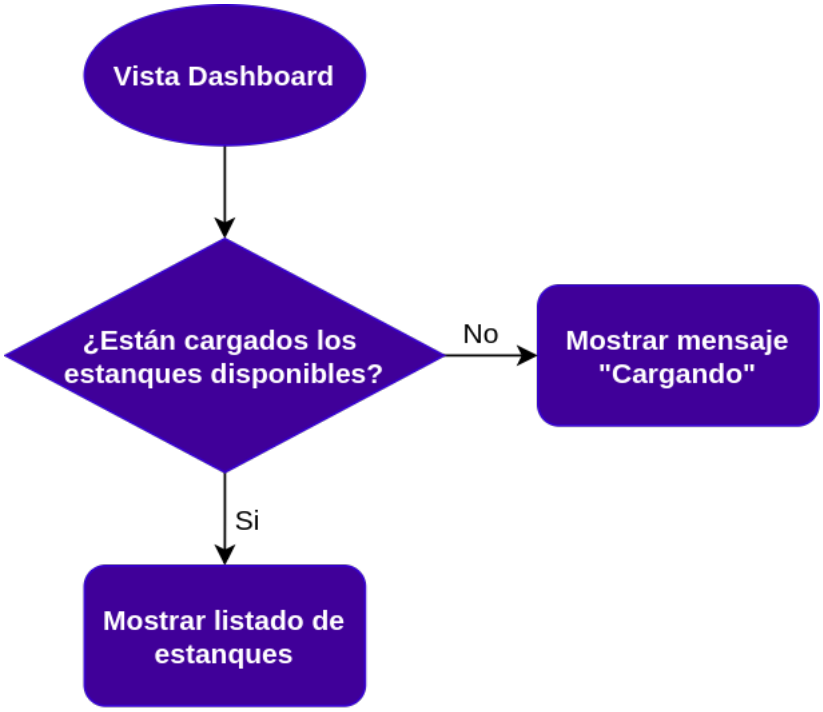



Figura 4.15: Diagrama de flujo de la lista de estanques en vista de dashboard.

Teniendo la lista de estanques desplegada, para obtener la predicción es necesario seleccionar alguno de éstos. Dicha acción gatilla una petición HTTP con método *GET* a la URL `http://localhost:8000/api/pond/predict/:pondId`, en donde `pondId` corresponde al identificador del estanque en la base de datos. Se incluyen headers

*'Content-Type': 'application/json'* y *'Authorization': 'Bearer* [*TOKEN*]*'*. Mientras se realiza la petición, se muestra un mensaje que indica que la predicción se está realizando. Al finalizar, se muestra la predicción del volumen en el estanque en las 24 horas del día a través de un gráfico. Dicho flujo se resume en la Figura 4.16.

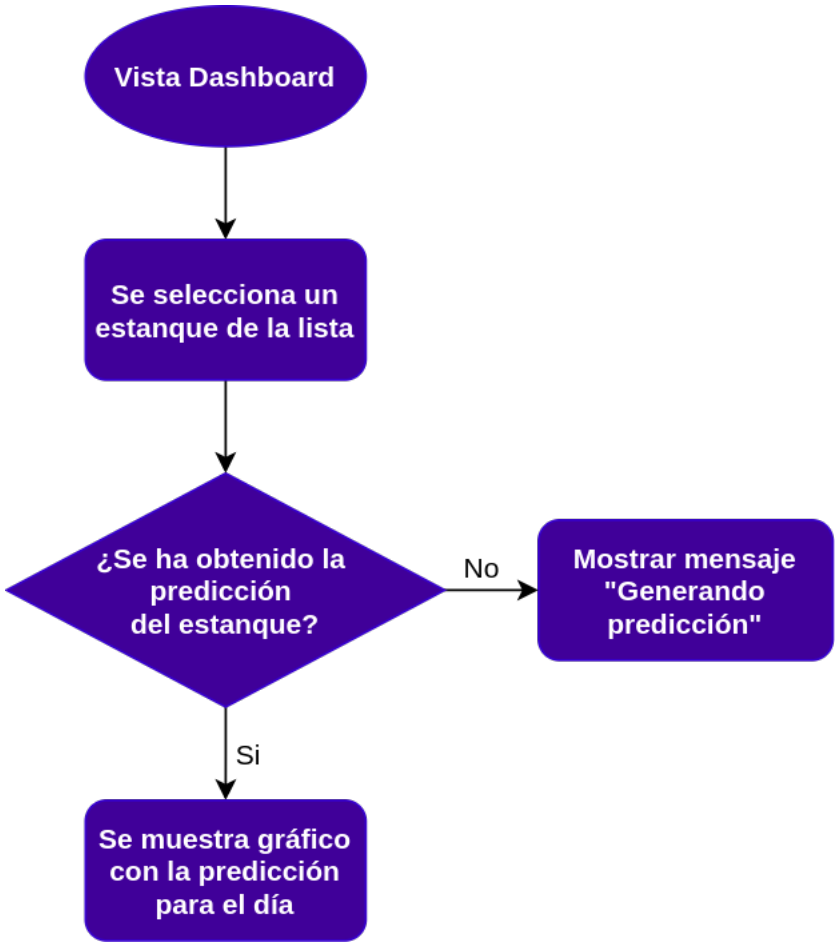



Figura 4.16: Diagrama de flujo de obtención de la predicción en vista de dashboard.

Al desear cerrar sesión, se debe ir al menú lateral y apretar el botón de *Cerrar Sesión*. Dicha acción gatilla el flujo descrito en 4.17, que consiste en la eliminación del token almacenado del navegador para luego redirigir al usuario a la vista de login.

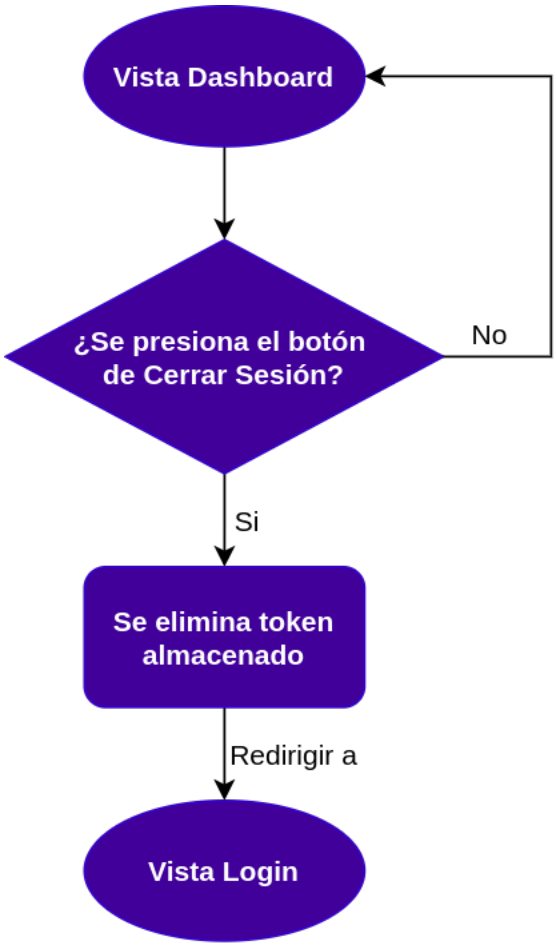



Figura 4.17: Diagrama de flujo para cerrar sesión

## 4.8. Resumen

A modo de resumen, se presenta la arquitectura del sistema en la Figura 4.18 donde se considera la integración del flujo de creación de modelos con la aplicación. En particular, se guardan tanto los archivos `.csv` y los modelos en un directorio de la aplicación para que sean utilizados al momento de generar las predicciones diarias gatilladas por solicitudes mediante la interfaz de usuario.

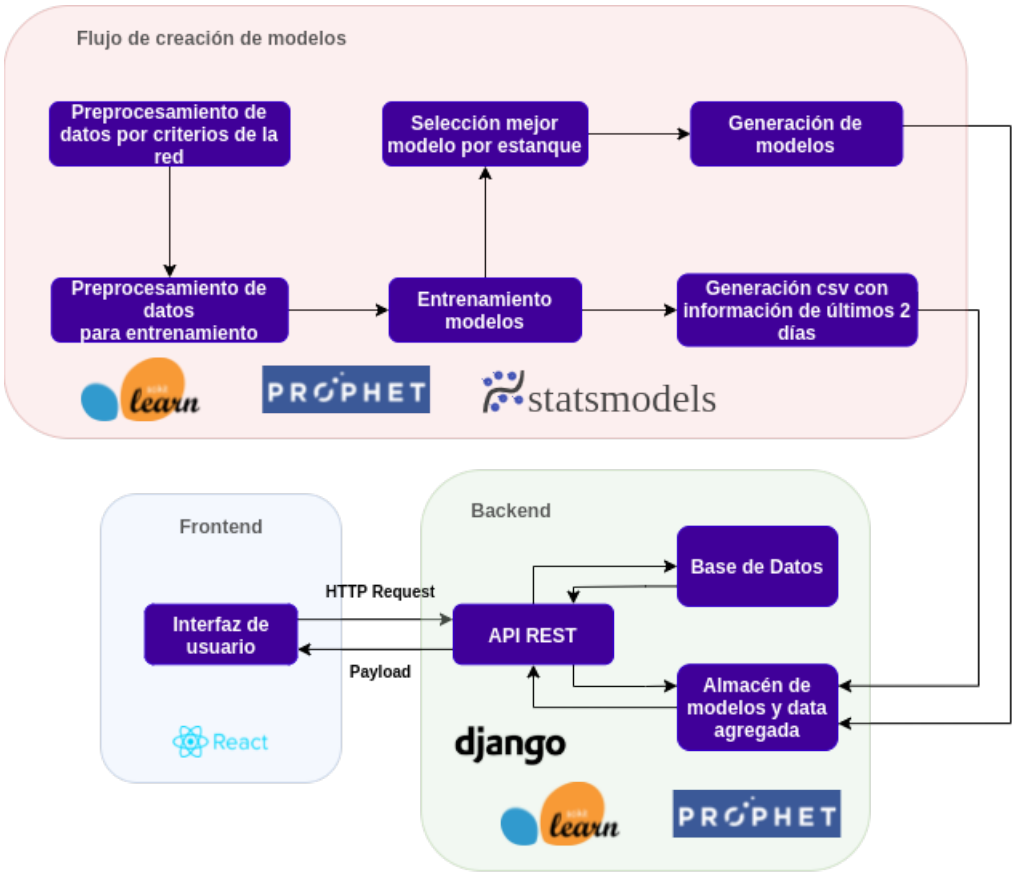



Figura 4.18: Arquitectura del sistema, incluyendo flujo de la aplicación y de creación de modelos.

Por otro lado, en el Cuadro 4.6 se compara la solución propuesta con los productos que ofrece la competencia, analizada previamente en el estado del arte.

Cuadro 4.6: Comparación entre Whome y la competencia.

| | **Whome** | **SIMULART** | **NOTUS SPA** | **DataQu** | **TaKaDu** |
|---|---|---|---|---|---|
| Simula operación del software | ✓ | ✓ | ✓ | | |
| Optimizar decisiones operativas | ✓ | ✓ | ✓ | ✓ | ✓ |
| Servicio en la nube | ✓ | | | ✓ | ✓ |
| Predicción en línea | ✓ | | | ✓ | ✓ |
| Gestión de eventos | | | | | ✓ |

# Capítulo 5

# Resultados

En esta sección se muestran los resultados de los modelos de predicción descritos en 4, también se presenta la plataforma desarrollada para realizar predicciones en línea utilizando los modelos entrenados.

## 5.1. Resultados métodos de predicción

Para todas la pruebas realizadas se considera un 70 % del dataset para entrenamiento, y el 30 % restante se utiliza para pruebas de rendimiento de los modelos, también se ajustó el muestreo de las mediciones para tener 1 medición por hora, se consideró la media de los valores de cada hora como valor final.

Se realizaron entrenamientos de 4 modelos distintos, todos con su configuración por defecto, para conocer como se compartan dependiendo del tamaño del dataset. Los escenarios a considerar son los siguientes:

- Escenario 1: dataset de 30 días.
- Escenario 2: dataset de 90 días.
- Escenario 3: dataset de 180 días.

Para efectos de medición del error se utilizó MAPE, y a partir de este valor se

obtienen las precisiones presentadas en los Cuadros 5.1, 5.2, 5.3. Considerar que el valor **S/I** corresponde a Sin Información, y hace relación a que en algún punto de la serie de tiempo el modelo no fue capaz de converger.

Cuadro 5.1: Precisión modelos escenario 1.

| Estanque | Modelo | | | |
|---|---|---|---|---|
| | GP | LR | Sarima | Prophet |
| Limache | 93.15 | 91.47 | 70.19 | 91.57 |
| La Calera | 96.20 | 95.75 | 78.68 | 94.98 |
| Villa Alemana | S/I | 89.11 | 61.91 | 89.16 |
| Quilpué | S/I | 87.10 | 52.28 | 87.08 |
| Concón | S/I | 82.63 | 68.93 | 82.61 |
| Valparaíso | S/I | 86.78 | 75.84 | 87.98 |
| Viña del Mar 1 | 94.25 | 93.91 | 76.52 | 94.78 |
| Viña del Mar 2 | S/I | 75.32 | 32.53 | 77.16 |
| Viña del Mar 3 | 89.82 | 89.92 | 71.57 | 83.94 |

Cuadro 5.2: Precisión modelos escenario 2.

| Estanque | Modelo | | | |
|---|---|---|---|---|
| | GP | LR | Sarima | Prophet |
| Limache | 71.23 | 74.94 | S/I | 77.61 |
| La Calera | 88.41 | 92.44 | 29.13 | 91.86 |
| Villa Alemana | 76.89 | 79.69 | S/I | 81.59 |
| Quilpué | S/I | 81.20 | S/I | 81.54 |
| Concón | S/I | 85.50 | 46.82 | 78.01 |
| Valparaíso | S/I | S/I | S/I | S/I |
| Viña del Mar 1 | 88.05 | 89.84 | 34.51 | 89.78 |
| Viña del Mar 2 | S/I | 77.84 | S/I | 69.54 |
| Viña del Mar 3 | 89.61 | 89.37 | 28.35 | 85.96 |

De los resultados obtenidos se evidencia que el escenario 1 es el más estable en términos de precisión. Además, a mayor tamaño de dataset se hace más difícil la convergencia de las predicciones, esto se nota en la cantidad de valores **S/I**, los cuales van en aumento a medida que se aumenta el dataset. Tomando esto en cuenta se decide trabajar con un dataset de tamaño fijo de 30 días.

Cuadro 5.3: Precisión modelos escenario 3.

| Estanque | Modelo | | | |
|---|---|---|---|---|
| | GP | LR | Sarima | Prophet |
| Limache | S/I | 79.76 | S/I | 79.35 |
| La Calera | S/I | 91.40 | 21.96 | 90.52 |
| Villa Alemana | S/I | S/I | S/I | 57.29 |
| Quilpué | S/I | S/I | S/I | S/I |
| Concón | 80.81 | 82.80 | 26.81 | 82.25 |
| Valparaíso | S/I | 83.00 | 43.28 | 84.03 |
| Viña del Mar 1 | 78.16 | 83.48 | 28.38 | 86.29 |
| Viña del Mar 2 | S/I | S/I | S/I | S/I |
| Viña del Mar 3 | S/I | 85.92 | 15.07 | 81.79 |

### 5.1.1. SARIMA

Se realizó una búsqueda de hiperparámetros considerando valores entre [0,3] para los parámetros $(p, d, q)x(P, D, Q)$, el parámetro $m$ se fija en 7 ya que se está trabajando con tendencia semanal. Se consideran las 3 mejores configuraciones para entrenar y probar los modelos.

- $(p, d, q)x(P, D, Q)m = (0, 0, 0)x(1, 0, 0)7$.
- $(p, d, q)x(P, D, Q)m = (0, 0, 0)x(0, 1, 0)7$.
- $(p, d, q)x(P, D, Q)m = (0, 0, 0)x(1, 1, 0)7$.

A continuación en el Cuadro 5.4, se presentan las precisiones sobre el dataset de prueba de cada configuración. En la Figura 5.1, se presentan las series de tiempo del dataset de prueba junto con la predicción obtenida por el modelo de configuración 2.

Cuadro 5.4: Precisión por estanque.

| Estanque | Configuración 1 | Configuración 2 | Configuración 3 |
|---|---|---|---|
| Limache | 70.19 | 70.98 | 70.97 |
| La Calera | 78.69 | 76.18 | 80.11 |
| Villa Alemana | 61.92 | 61.99 | 57.09 |
| Quilpué | 52.28 | 55.08 | 49.64 |
| Concón | 68.94 | 69.63 | 69.44 |
| Valparaíso | 75.85 | 78.95 | 73.81 |
| Viña del mar 1 | 76.52 | 76.15 | 78.51 |
| Viña del mar 2 | 32.53 | 40.64 | 28.98 |
| Viña del mar 3 | 71.57 | 68.42 | 71.66 |
| | | | |
| Promedio | 65.39 | 66.45 | 64.47 |

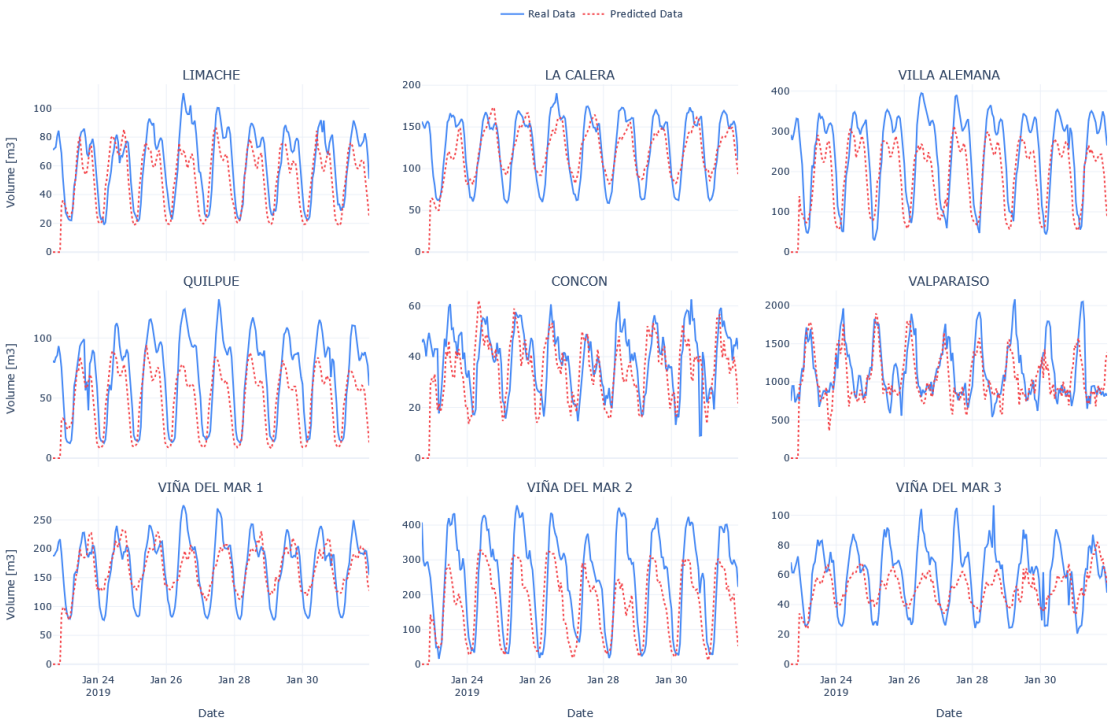



Figura 5.1: Series de tiempo real v/s predicción, escenario 1 configuración 2.

### 5.1.2. Prophet

Como Prophet es un framework que se encarga de buscar las mejores configuraciones de forma automática, no hay necesidad de buscar parámetros que se ajusten

a los datos. Para entrenar el modelo la única configuración que se hace es definir el parámentro *weekly_seasonality* como verdadero para considerar que los datos poseen tendencia semanal. A continuación en el Cuadro 5.5, se presenta la precisión sobre el dataset de prueba, y en la Figura 5.2, se muestras las series de tiempo junto con la predicción obtenida.

Cuadro 5.5: Precisión por estanque.

| **Estanque** | **Configuración 1** |
|---|---|
| Limache | 91.57 |
| La Calera | 94.98 |
| Villa Alemana | 89.16 |
| Quilpué | 87.08 |
| Concón | 82.61 |
| Valparaíso | 87.98 |
| Viña del mar 1 | 94.78 |
| Viña del mar 2 | 77.16 |
| Viña del mar 3 | 83.94 |

| | |
|---|---|
| Promedio | 87.70 |

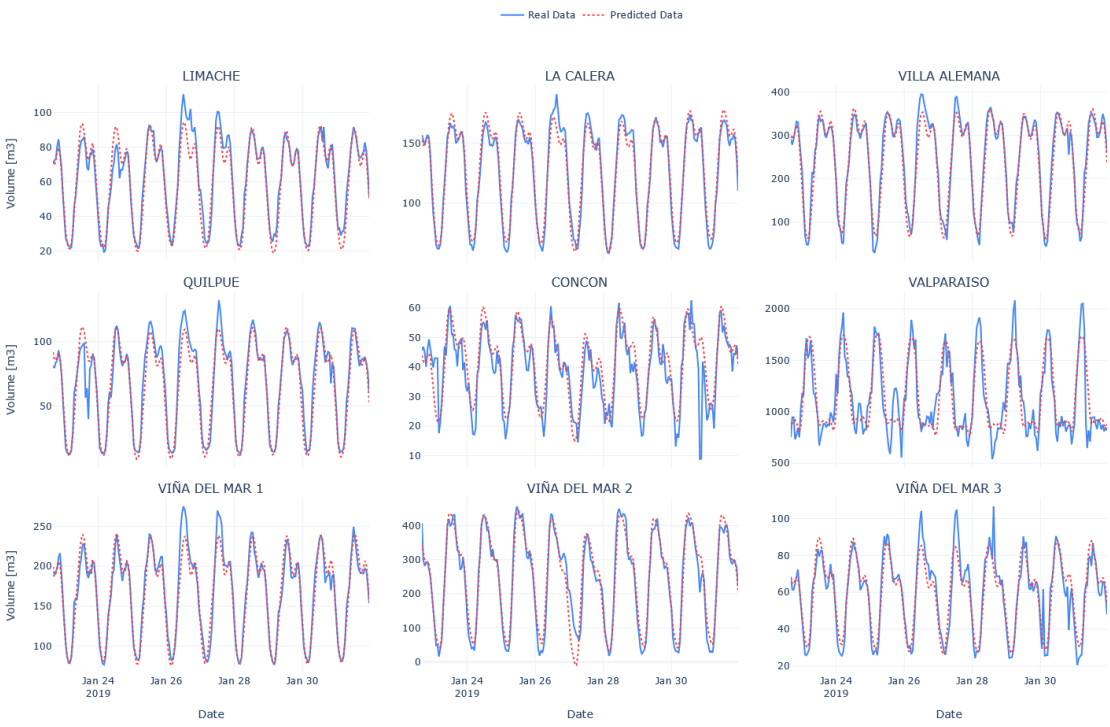



Figura 5.2: Series de tiempo real v/s predicción, escenario 1 configuración 1.

### 5.1.3. Regresión Lineal

Para el modelo de regresión lineal se amplió el dataset con información de la hora de medición, día de la semana y un indicador si es fin de semana. Además de la regresión lineal simple se llevaron a cabo entrenamientos considerando regularización Ridge y Lasso. A continuación en el Cuadro 5.6, se presenta la precisión sobre el dataset de prueba, y en la Figura 5.3, se presentan las series de tiempo del dataset de prueba junto con la predicción obtenida de una regresión lineal sin regularización.

Cuadro 5.6: Precisión por estanque.

| Estanque | Configuración 1 | Configuración 2 | Configuración 3 |
|---|---|---|---|
| Limache | 91.47 | 91.45 | 90.74 |
| La Calera | 95.75 | 95.74 | 95.40 |
| Villa Alemana | 89.11 | 89.12 | 88.75 |
| Quilpué | 87.10 | 87.06 | 86.03 |
| Concón | 82.63 | 82.63 | 81.92 |
| Valparaíso | 86.78 | 86.80 | 86.93 |
| Viña del mar 1 | 93.91 | 93.93 | 93.96 |
| Viña del mar 2 | 75.32 | 75.31 | 74.83 |
| Viña del mar 3 | 89.92 | 89.93 | 89.73 |
| | | | |
| Promedio | 88.01 | 88.00 | 87.59 |

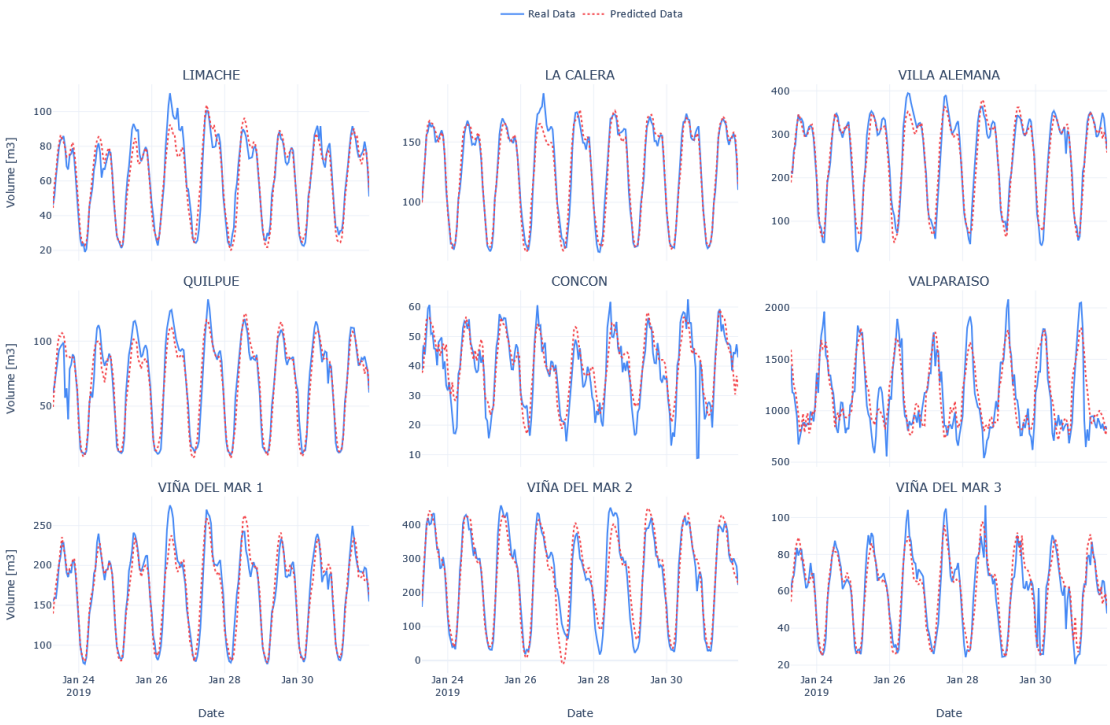



Figura 5.3: Series de tiempo real v/s predicción, escenario 1 configuración 1.

### 5.1.4. Procesos Gaussianos

Para el modelo basado en procesos Gaussianos también se amplió el dataset con información de la hora de medición, día de la semana y un indicador si es fin de

semana. Se utilizaron 4 configuraciones de hiperparámetros:

- $\sigma_y = 4{,}0$ y RBF como función de kernel.
- $\sigma_y = 1\epsilon^{-10}$ y RBF como función de kernel, con ruido.
- $\sigma_y = 1\epsilon^{-10}$ y Matern como función de kernel, con ruido.
- $\sigma_y = 1\epsilon^{-10}$ y Rational Quadratic como función de kernel, con ruido.

A continuación en el Cuadro 5.7, se presenta la precisión sobre el dataset de prueba, y en la Figura 5.4, se presentan las series de tiempo del dataset de prueba junto con la predicción obtenida utilizando la tercera configuración de parámetros (mayor precisión promedio).

Cuadro 5.7: Precisión por estanque.

| **Estanque** | **Configuración 1** | **Configuración 2** | **Configuración 3** | **Configuración 4** |
|---|---|---|---|---|
| Limache | 93.15 | 93.20 | 93.59 | 93.30 |
| La Calera | 96.20 | 97.08 | 97.10 | 97.08 |
| Villa Alemana | S/I | 91.16 | 91.39 | 91.49 |
| Quilpué | S/I | 89.92 | 90.40 | 90.19 |
| Concón | S/I | 84.58 | 84.97 | 85.14 |
| Valparaíso | S/I | 87.51 | 87.51 | 87.46 |
| Viña del mar 1 | 94.25 | 95.58 | 95.68 | 95.64 |
| Viña del mar 2 | S/I | 83.75 | 85.53 | 86.00 |
| Viña del mar 3 | 89.82 | 91.37 | 91.81 | 91.63 |
| Promedio | 93.35 | 90.46 | 90.89 | 90.88 |

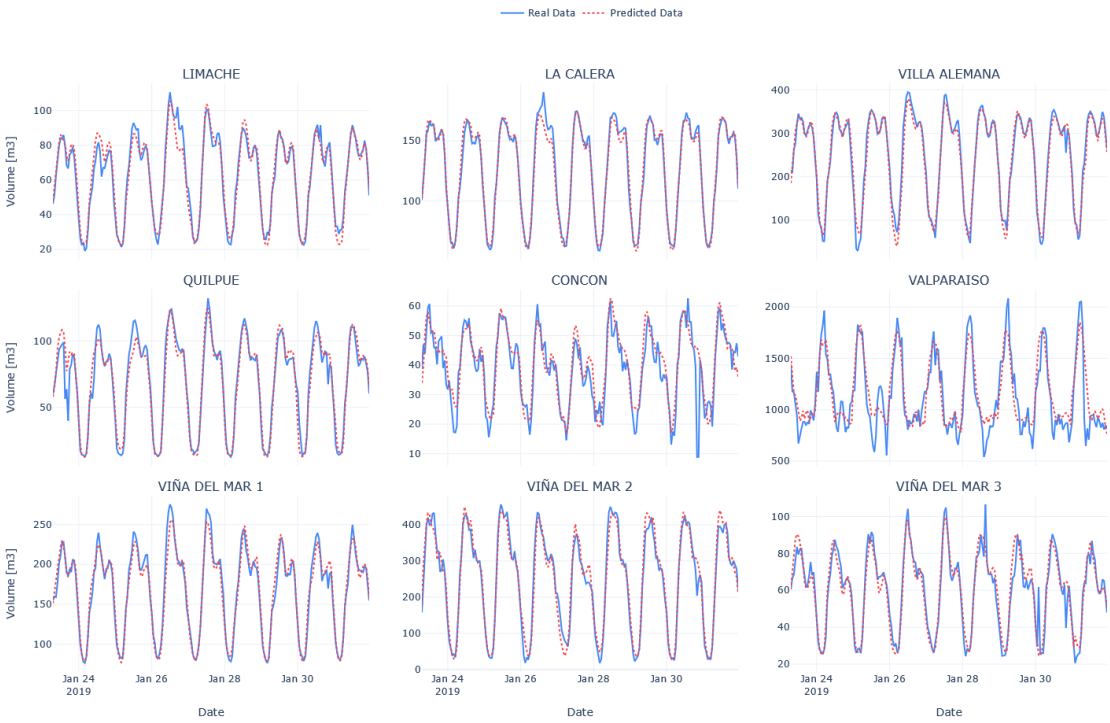



Figura 5.4: Series de tiempo real v/s predicción, escenario 1 configuración 3.

## 5.2. Aplicación

A continuación se muestran los resultados asociados a la aplicación del sistema, esta posee un login que en el caso de que se ingresen credenciales correctas, se redirige a una vista de dashboard que dispone de un listado de estanques disponibles para obtener la predicción diaria. Al seleccionar uno de éstos, se procede a calcular la predicción para luego mostrarla a través de un gráfico.

### 5.2.1. Vista login

Se muestra en la Figura 5.10 la vista de login, que corresponde a la vista inicial de la aplicación. En ésta se completan las credenciales con un nombre de usuario y contraseña. En el caso de que las credenciales sean correctas, se redirige al usuario a la vista de dashboard.

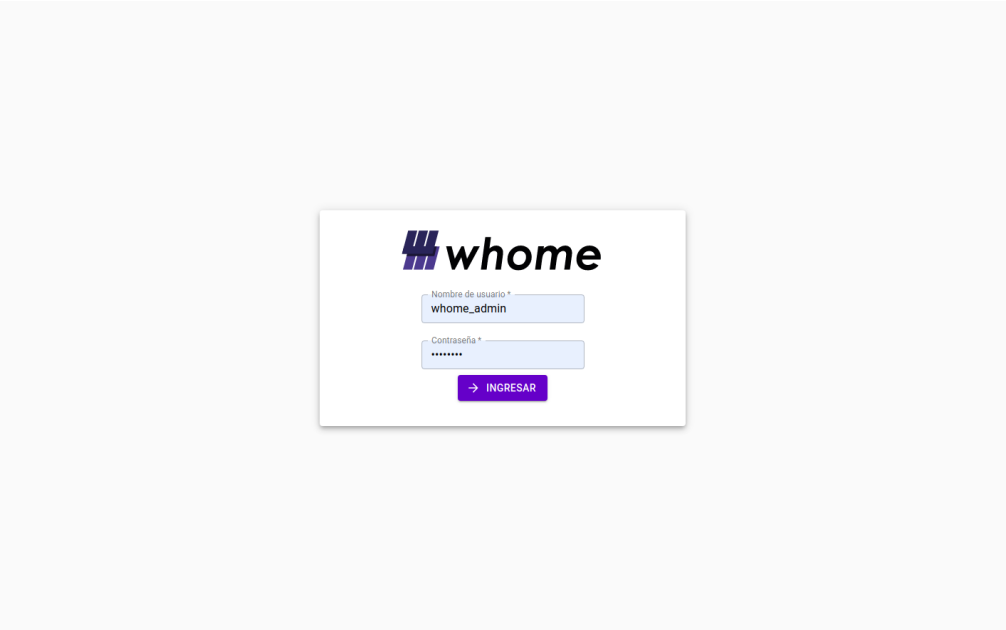



Figura 5.5: Vista de login.

### 5.2.2. Vista dashboard

En esta vista se muestran tanto los estanques disponibles como los resultados de la predicción diaria al momento de seleccionar un estanque. Al cargar la vista, se cargan los estanques disponibles mostrando un mensaje como se muestra en la Figura 5.6. Al terminar la carga, se aparece el listado como se muestra en la Figura 5.7.

Luego, para obtener la predicción diaria de un estanque se selecciona alguno de los estanques de la lista. La acción anterior gatilla el cálculo de la predicción, mostrando un mensaje notificando el estado de carga como se muestra en la Figura 5.8. En la Figura 5.9 se puede visualizar un gráfico con los resultados al terminar el cálculo.

Finalmente, en la Figura 5.10 se muestra el menú lateral expandido, en donde se muestra el botón de Cerrar Sesión que al apretarlo redirecciona a la vista de login luego de haber realizado las operaciones necesarias para cerrar sesión.

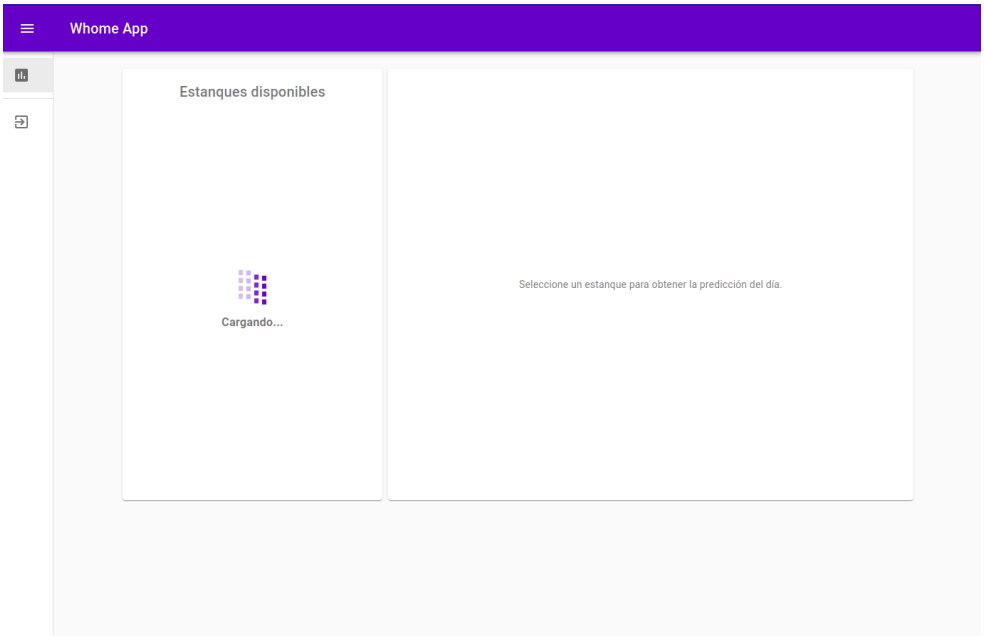



Figura 5.6: Vista de dashboard al estar cargando el listado de estanques.

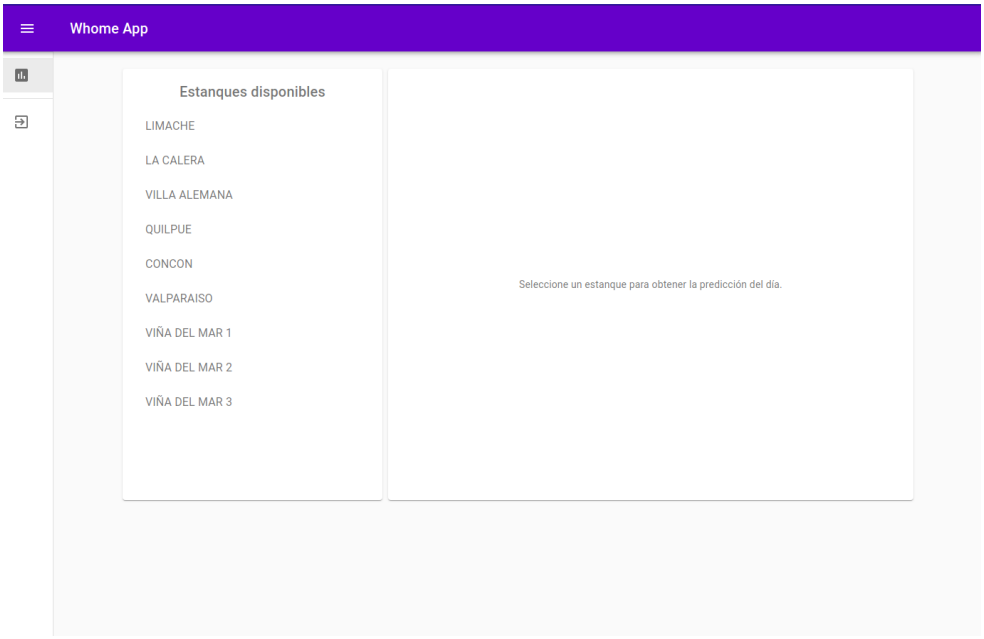



Figura 5.7: Vista de dashboard con lista de estanques disponible.

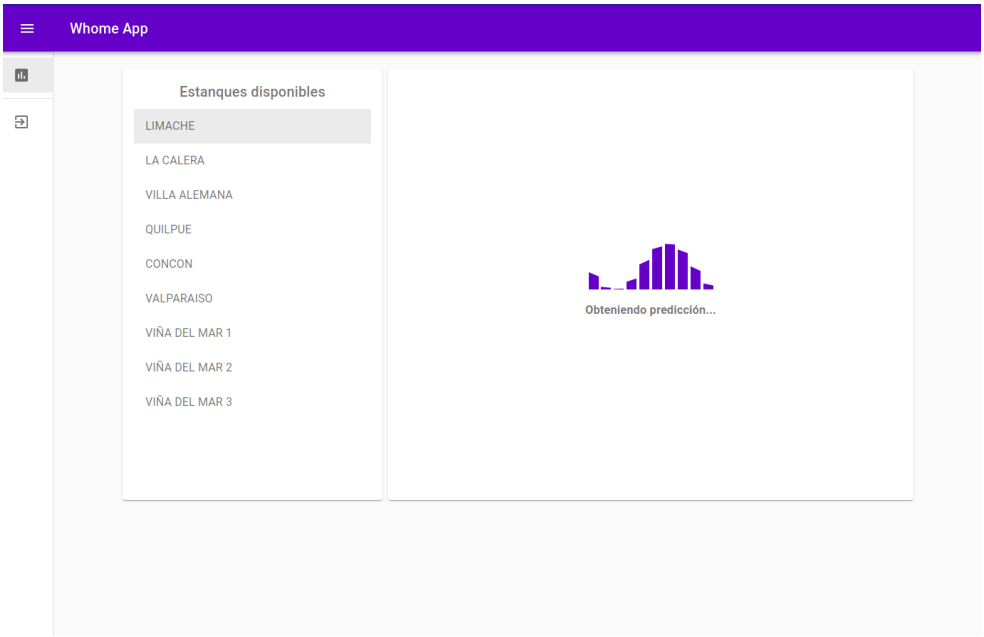



Figura 5.8: Vista de dashboard al estar obteniendo la predicción de un estanque seleccionado.

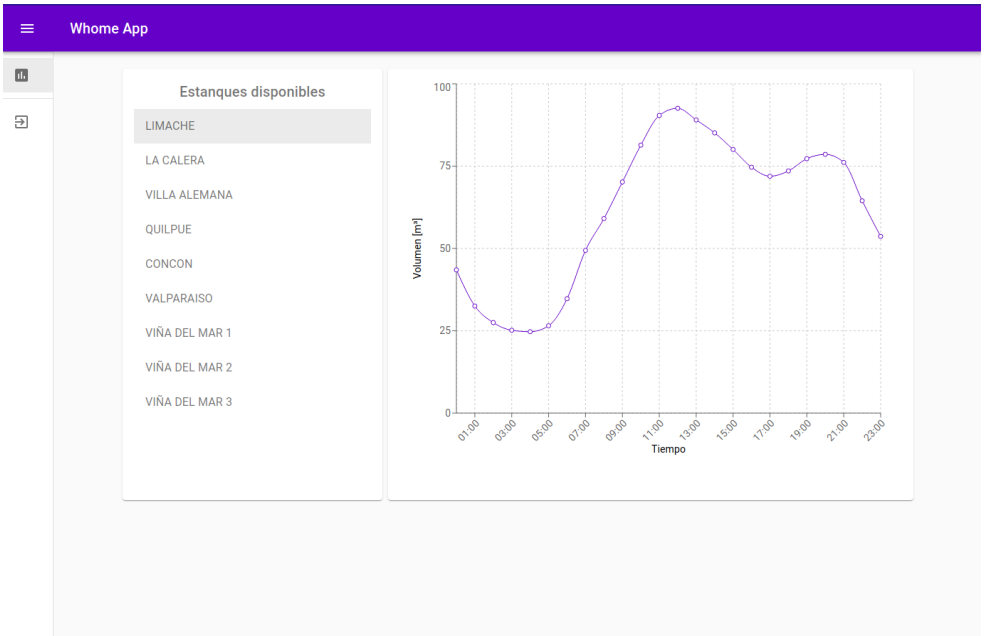



Figura 5.9: Vista de dashboard al obtener la predicción de un estanque seleccionado.

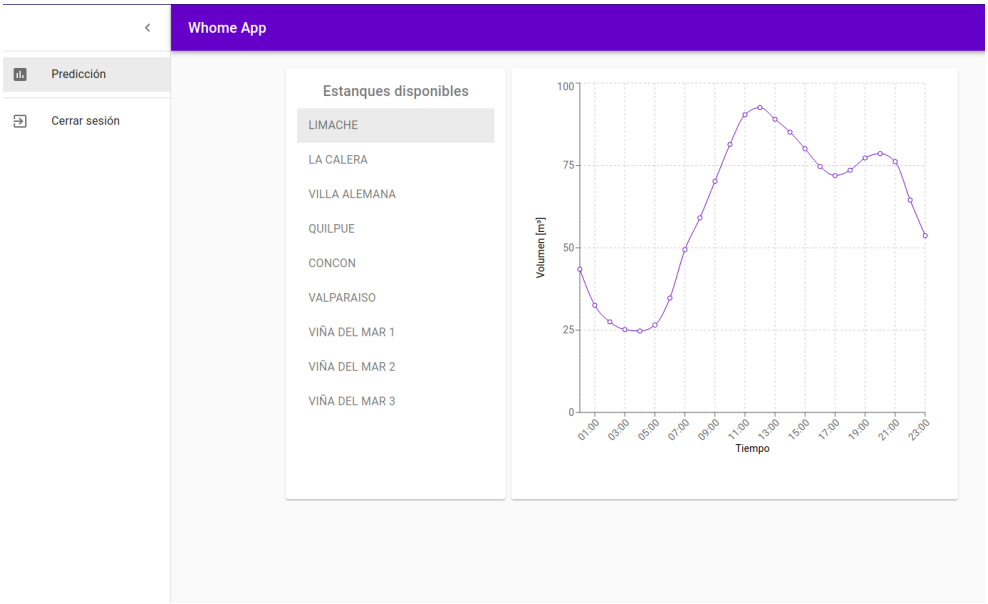



Figura 5.10: Vista de dashboard al expandir el menú lateral.

# Capítulo 6

# Conclusiones y Trabajos futuros

Este trabajo presenta una solución sobre predicción de demanda de agua potable a corto plazo. La propuesta es la creación de una aplicación que permite al usuario seleccionar un estanque, y mediante una API se proporciona información de la demanda del día siguiente.

El desempeño de modelos SARIMA, independiente de la configuración de hiperparámetros, es regular. Si bien el desempeño no es del todo malo, para el rubro en donde se usará el sistema no es tolerable errores de esta magnitud. Los resultados con Prophet son positivos, en general el desempeño es bueno (en promedio 87.70 %) y la implementación es la más simple de todas.

Con respecto a los modelos de predicción tradicionales, ya sea regresión lineal o basado en procesos Gaussianos, ambos tienen un muy buen desempeño (por encima de Prophet) sin siquiera realizar una búsqueda de parámetros que se ajusten a cada estanque.

Con respecto a la aplicación, se logra obtener la predicción diaria de cada estanque de la red reducida desarrollada mediante una interfaz de usuario simple, junto con control de acceso mediante credenciales. Lo anterior le permite a la empresa tomar acciones en base a la demanda que tenga cada estanque, el cual beneficia en términos de costo operacional. Bajo esta linea, queda como trabajo futuro evaluar cuantitativa-

mente la reducción de costos.

Por otro lado, queda como trabajo futuro el realizar una búsqueda de parámetros ideales para cada modelo y cada estanque, de esta forma se podría alcanzar precisiones aún mayores a las obtenidas. En cuanto a la aplicación, seria ideal agregar una funcionalidad para entrenar los modelos a medida que se añada información nueva, de esta manera se puede pasar a un sistema cuyos modelos se vayan actualizando automáticamente día a día.

Por último, queda pendiente la integración del sistema presentado con un sistema que tome la información provista, realice una optimización de los recursos en base a la demanda de agua, plantas productoras, y costo de las plantas productoras. Esto con el objetivo de entregar una planificación del día a los operarios.

# Bibliografía


[1] Wolrd Resources Institute. https://www.wri.org/insights/domestic-water-use-grew-600-over-past-50-years.

[2] El Dínamo. https://www.eldinamo.cl/ambiente/2020/04/04/agua-y-sequia-cuales-son-las-industrias-que-mas-emplean-este-recurso-y-que-estan-haciendo-para-darle-un-uso-mas-eficiente/, Abril 2020.

[3] Esval. https://www.esval.cl/.

[4] Natalia Rodríguez Aedo. PRONOSTICO DE DEMANDA DE AGUA POTABLE MEDIANTE REDES NEURONALES. http://hdl.handle.net/11673/13205, Junio 2016.

[5] Ministerio de Salud. https://dipol.minsal.cl/departamentos-2/salud-ambiental/, 2021.

[6] Reporte de Sostenibilidad, Esval. http://portal.esval.cl/wp-content/uploads/2020/10/Reporte-de-Sostenibilidad-Esval-2019.pdf, Octubre 2019.

[7] Norma Calidad del Agua Potable. https://aguaslosmaitenes.cl/documentos/agua_potable/Normas%20NCh%20409%20Calidad%20y%20Muestreo%20del%20Agua%20Potable%20EEO.pdf, Enero 2007.

[8] Superintendencia de Servicios Sanitarios, Aguas No Facturadas. https://www.siss.gob.cl/586/articles-17573_recurso_1.pdf, 2018.

[9] Zhang S, Yang J, Wan Z, Yi Y. Multi-Water Source Joint Scheduling Model Using a Refined Water Supply Network: Case Study of Tianjin. Water. 2018; 10(11):1580. https://doi.org/10.3390/w10111580, 2018.

[10] Yamout, G., El-Fadel, M. An Optimization Approach for Multi-Sectoral Water Supply Management in the Greater Beirut Area. Water Resour Manage 19, 791–812 (2005). https://doi.org/10.1007/s11269-005-3280-6, 2005.

[11] Wikipedia. https://es.wikipedia.org/wiki/Beirut.

[12] Tom M Mitchell et al. Machine learning. 1997.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] Shah, S., Hosseini, M., Miled, Z. B., Shafer, R., Berube, S. A Water Demand Prediction Model for Central Indiana. Thirtieth AAAI Conference on Innovative Applications on Artificial Intelligence (IAAI-18). https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16511, 2018.

[15] Lopez Farias R, Puig V, Rodriguez Rangel H, Flores JJ. Multi-Model Prediction for Demand Forecast in Water Distribution Networks. Energies. 2018; 11(3):660. https://doi.org/10.3390/en11030660, 2018.

[16] A. Antunes, A. Andrade-Campos, A. Sardinha-Lourenço, M. S. Oliveira. Short-term water demand forecasting using machine learning techniques. Journal of Hydroinformatics, vol. 20, nº 6, pp. 1343–1366. https://doi.org/10.2166/hydro.2018.163, Agosto 2018.

[17] Kühnert C, Gonuguntla NM, Krieg H, Nowak D, Thomas JA. Application of LSTM Networks for Water Demand Prediction in Optimal Pump Control. Water. 2021; 13(5):644. https://doi.org/10.3390/w13050644, 2021.

[18] Praveen Vijai and P Bagavathi Sivakumar. Performance comparison of techniques for water demand forecasting. *Procedia Computer Science*, 143:258–266, 2018. 8th International Conference on Advances in Computing Communications (ICACC-2018).

[19] SIMULART. http://www.simulart.cl/.

[20] NOTUS SPA. https://www.notus.cl/.

[21] Dictuc. https://www.dictuc.cl/.

[22] DataQu. https://dataqu.ai/.

[23] TaKaDu. https://www.takadu.com/.

[24] SCADA iFIX. https://www.opertek.com/software/scada-ifix/.

[25] Stack Overflow 2020 Developer Survey. https://insights.stackoverflow.com/survey/2020.

[26] React. https://reactjs.org/.

[27] Angular. https://angular.io/.

[28] Vue JS. https://vuejs.org/.

[29] Django. https://www.djangoproject.com/.