Cabado, Franco Leonel

# Sistema informático de monitoreo y seguimiento de la calidad del agua en arroyos

2020

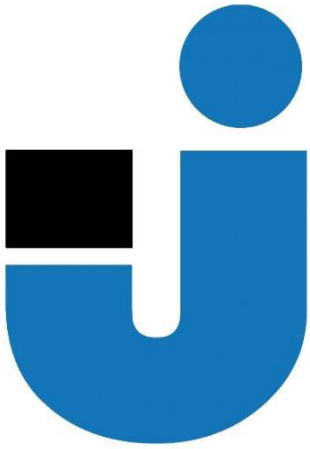
*Instituto: Ingeniería y Agronomía*

*Carrera: Ingeniería en Informática*

Universidad Nacional Arturo Jauretche

**Instituto de Ingeniería y Agronomía**

**Ingeniería en Informática**



**TRABAJO FINAL DE LA PRÁCTICA PROFESIONAL SUPERVISADA**

# *Sistema informático de monitoreo de la calidad del agua en arroyos (SICA)*

**Estudiante:**
***Franco Leonel Cabado***

**Tutores:**

***Prof. Lic. Carolina Kelly***

***Prof. Dr. Marcelo Cappelletti***

***Prof. Mg. Jorge Osio***

**Buenos Aires, 2020**

# PRÁCTICA PROFESIONAL SUPERVISADA (PPS)
## Sistema informático de monitoreo de la calidad del agua en arroyos
## Informe de Final

**DATOS DEL ESTUDIANTE**

Apellido y Nombres: Franco Leonel Cabado

DNI Nº: 37878375

Nº de Legajo: 10820

Correo electrónico: leonelcabado7@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 41

PPS enmarcada en artículo 4 de la Resolución (CS) 103/16.

**DOCENTE SUPERVISOR**

Apellido y Nombres: Prof. Dr. Marcelo Cappelletti

Correo electrónico: mcappelletti@unaj.edu.ar

**DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS DE LA UNAJ**

Apellido y Nombres: Prof. Lic. Carolina Kelly

Correo electrónico: kellygcarolina@gmail.com

**DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS**

Nombre o Razón Social: Universidad Nacional Arturo Jauretche

Dirección: Av. Calchaquí 6200, Florencio Varela, Buenos Aires

Teléfono: +54 11 4275-6100

Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, Instituto de Ingeniería y Agronomía

**TUTOR DE LA ORGANIZACIONAL**

Apellido y Nombres: Prof. Mg. Jorge Osio

Correo electrónico: josio@unaj.edu.ar

**FIRMA DEL COORDINADOR DE LA CARRERA**

# Sistema Informático para la Calidad del Agua

# (SICA)

## Índice

## 1. Introducción

La presente Práctica Profesional Supervisada (PPS) consiste en la definición, el diseño, el desarrollo y la implementación de un sistema integral y escalable de monitoreo y seguimiento de la calidad del agua en arroyos con el fin de elaborar planes de contingencia adecuados, en caso de ser necesarios, y reducir, así, el riesgo de infecciones y/o enfermedades que, debido a su contaminación, esta podría llegar a generar.

La propuesta del presente trabajo surgió a partir de la observación del proceso de recolección de datos, sobre la calidad del agua en arroyos, que existe actualmente: primero, se introduce una sonda en el arroyo y la información obtenida de los sensores se almacena en una tarjeta denominada *Digital Secure* (Seguro Digital o SD, un dispositivo en formato de tarjeta de memoria para implementos portátiles). Este proceso tiene la desventaja de que se debe esperar varios días hasta retirar la sonda y poder acceder a los datos, por lo que resulta poco práctico y ágil para realizar la visualización y el análisis de la información. Por ello, se pensó en sistematizar y efectivizar el proceso mediante una aplicación web, a fin de obtener la información que se va registrando de una manera más rápida y eficiente.

La aplicación llevará el nombre de "Sistema Informático para la Calidad del Agua" (de acuerdo con sus siglas y, de ahora en más, SICA) y se alojará en los servidores de la Universidad Nacional Arturo Jauretche para su funcionamiento.

### 1.1 Objetivos

El presente trabajo tiene entonces una serie de objetivos de carácter general, vinculados al desarrollo de la aplicación; es decir, los siguientes objetivos son los que dicha aplicación debe poder realizar:

- Elaboración de gráficas del estado actual de la calidad del agua.
- Confección de gráficas de estados históricos de la calidad del agua, a fin de brindar un índice que dé cuenta del mejoramiento (o no) de los planes de contingencias generados.
- Realización de reportes sobre la información registrada.
- Implementación de "alertas" y "notificaciones" del estado de los sensores.

- Ejecución de un listado detallado de las "localizaciones" donde se encuentra activo el sistema.
- Implementación del sistema completo en los servidores de la Universidad Nacional Arturo Jauretche mediante contenedores de *software*.

### 1.2 Tareas a realizar

Como próximo paso, se presentará una metodología de desarrollo del diseño para poder trabajar de forma eficiente y ordenada, sin perder ningún aspecto en cuanto a las funcionalidades que deberá tener el sistema. La metodología denominada "en cascada" es la que se utilizará para llevar adelante su desarrollo. Se eligió dicha metodología, porque posibilita una codificación del *software* rápida y muy estructurada. Cada fase del proyecto se encuentra bien definida y delimitada, lo que facilita el entendimiento de sus funcionalidades.

Las fases que presenta la metodología elegida se establecen en la ilustración 1:


*Ilustración 1: Ionos, guía digital (2019). Metodología de desarrollo del proyecto. [Diagrama].*
(Recuperado de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/)

Básicamente, la imagen muestra un modelo lineal de diseño de *software* que emplea un proceso de diseño secuencial. El desarrollo fluye secuencialmente desde el punto inicial hasta

el punto final, con varias etapas o fases diferenciadas: en este caso, son las etapas de análisis, diseño, implementación, verificación y mantenimiento.

El énfasis de la metodología en cascada se pone en la planificación del proyecto y, por lo tanto, antes de comenzar cualquier tipo de desarrollo, es necesario que tanto la visión como el plan estén claros, debido a que el método en cascada requiere un amplio esfuerzo de preparación previa. Esto permite, en primer lugar, comenzar con el *software* con bastante rapidez; en segundo lugar, estimar calendarios y presupuestos con mayor precisión; en tercer lugar, lograr un nivel de satisfacción del cliente más elevado que otros enfoques; y, finalmente, establecer procesos de desarrollo seguros y sostenibles, debido a que su documentación puede ser retomada por cualquier otro desarrollador.

### 1.3 Cronograma de trabajo

| TAREAS | PERIODOS (MESES) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Julio | Agosto | Septiembre | Octubre | Noviembre | Diciembre | Enero | Febrero | Marzo | Abril | Mayo | Junio |
| Relevamiento | | | | | | | | | | | | |
| Modelados y diseño del proceso actual | | | | | | | | | | | | |
| Validación del flujo del proceso actual | | | | | | | | | | | | |
| Toma y clasificación de requerimientos | | | | | | | | | | | | |
| Validación de requerimientos | | | | | | | | | | | | |
| Desarrollo | | | | | | | | | | | | |
| Modelado y diseño de funcionalidades del sistema | | | | | | | | | | | | |
| Presentar el plan de informe de la PPS | | | | | | | | | | | | |
| Desarrollo de la parte lógica del sistema (BACKEND) | | | | | | | | | | | | |
| Desarrollo de la parte gráfica del sistema (FRONTEND) | | | | | | | | | | | | |
| Implementación de aplicativo | | | | | | | | | | | | |
| Presentar avance del informe de la PPS | | | | | | | | | | | | |
| Pruebas y validaciones | | | | | | | | | | | | |
| Validaciones finales y conclusiones | | | | | | | | | | | | |
| Presentación PPS | | | | | | | | | | | | |

*Ilustración 2: Cronograma de trabajo*
(Imagen propia, basada en la práctica)

En lo que sigue, se definirá y desarrollará el sistema completo, el diseño de las tecnologías implicadas, las funcionalidades e implementación del aplicativo a nivel productivo.

## 2. Desarrollo

### 2.1 Definición y desarrollo del sistema completo

A la hora de plantear un nuevo proyecto, definir bien sus tiempos y otorgarle una estructura adecuada son dos acciones de máxima prioridad y, tal como se mencionó, es el aspecto más importante de la metodología de trabajo elegida.

Como primer paso entonces, se realizó un análisis y una comprensión del sistema de sensores, el que fue investigado en primera instancia. De dicho análisis se obtuvieron los siguientes resultados:

### 2.1.1 Sistema sensorial

Consiste en un sistema de monitoreo hídrico que almacena, en una tarjeta SD, valores intrínsecos del agua y detecta si se encuentran en un rango que es nocivo para la salud, poniendo en evidencia su potabilidad. Básicamente, el sistema sensorial se define como una sonda multiparamétrica que mide de la calidad del agua en arroyos. Si bien existen múltiples valores que permiten determinar la contaminación en arroyos, los más relevantes, y que tiene en consideración este sistema, son la turbidez, la temperatura y la conductividad. Los sensores utilizados para obtener estos parámetros consisten en circuitos integrados (conjunto de conductores y componentes eléctricos interconectados por los que fluye corriente eléctrica) y que poseen el siguiente *hardware* y *software* embebidos:

### 2.1.2 Descripción del funcionamiento del *hardware* embebido

- **Microcontrolador Atmega328p ATMEL**

  Es un circuito integrado programable, capaz de ejecutar instrucciones guardadas en la memoria incorporada: se encarga de muestrear los parámetros de temperatura, turbidez y salinidad (conductividad) de manera directa y tomar el registro del tiempo de cada muestra. Adicionalmente, como ya se refirió, se encarga de almacenar la información en un archivo dentro de la memoria SD.

- **Sensor de temperatura sumergible (DS18B20)**

  Es un termistor (componente resistivo a la corriente eléctrica cuyo valor varía en función de la temperatura) con salida digital, que es capaz de sensar un amplio rango desde -55°C a 125°C.

- **Sensor de turbidez (SEN0189)**

  Mide las partículas sostenidas en el líquido mediante el envío de un haz infrarrojo emitido desde un lado del sensor y analiza la refracción en el lado opuesto al que es emitido (se emite desde un lado y se refracta en el otro). La medida se obtiene

normalizada en NTU (unidad en la que se mide la turbidez de un fluido o la presencia de partículas en suspensión en el agua).

- **Sensor de conductividad**

  Se implementó con un LM324, que es un circuito integrado compuesto de cuatro circuitos Amplificadores Operacionales (AO) que, a su vez, conforman las etapas del circuito. Mediante la utilización de dos electrodos sumergidos en el agua, lee un valor de tensión que oscila entre los 0mV y 250mV, según el nivel de resistividad.

- **Módulo RTC DS3231 (lectura de tiempo)**

  Es el encargado de llevar el registro de la fecha y la hora para tener la referencia del momento en que se realizaron las respectivas lecturas de los sensores.

Por último, y para finalizar la descripción de los componentes imprescindibles del sistema sensorial, el diseño incluye un batería de Litio-Ion de 3,6V a 2880mAh, que se encarga de proveer la energía necesaria para que funcione el sistema del *hardware* embebido.

### 2.1.3 Descripción del funcionamiento del *software* embebido

La lógica de ejecución siguió los siguientes pasos:

- Se decidió mantener la batería en estado de reposo por razones de sostenibilidad y eficiencia, para capitalizar, en su máximo posible, su vida útil. De esta forma, solo se activa con el uso de interrupciones generadas mediante la manipulación de los circuitos integrados del sistema.
- Cuando se produce la interrupción, se procede a la lectura de la señal emitida por los sensores y, además, se registra la fecha y hora de la muestra.
- Se almacenan los datos obtenidos de la lectura sensorial en un archivo de texto en la memoria SD.
- En simultáneo a la lectura de los sensores, también se obtiene un valor del estado de la batería: si se encuentra por debajo de los 2.9V, el sistema ejecuta un modo de protección cortando todo consumo de energía del sistema.
- Por último, el dispositivo entra en estado de reposo nuevamente, a la espera de otra interrupción.

A continuación, se puede observar el archivo de texto que genera el sistema:

| | | | |
|---|---|---|---|
| 18.56°C | 2.00NTU | 2/7/2019 | 16:1:10 |
| 18.56°C | 2.00NTU | 2/7/2019 | 16:1:22 |
| 18.69°C | 2.00NTU | 2/7/2019 | 16:2:2 |
| 20.50°C | 2.00NTU | 10/7/2019 | 15:2:6 |
| 20.69°C | 2.00NTU | 10/7/2019 | 15:2:34 |
| 20.69°C | 2.00NTU | 10/7/2019 | 15:3:47 |
| 20.62°C | 2.00NTU | 10/7/2019 | 15:5:2 |
| 21.31°C | 3.00NTU | 10/7/2019 | 15:11:53 |
| 21.31°C | 3.00NTU | 10/7/2019 | 15:13:52 |
| 21.44°C | 3.00NTU | 10/7/2019 | 15:18:58 |
| 21.44°C | 3.00NTU | 10/7/2019 | 15:22:54 |
| 21.31°C | 3.00NTU | 10/7/2019 | 15:23:49 |
| 21.37°C | 3.00NTU | 10/7/2019 | 15:25:37 |
| 21.37°C | 3.00NTU | 10/7/2019 | 15:28:58 |
| 21.37°C | 2.00NTU | 10/7/2019 | 15:30:51 |
| 21.50°C | 2.00NTU | 10/7/2019 | 15:33:49 |

*Ilustración 3: Salida de la sonda paramétrica*
(Imagen propia, basada en la práctica)

### 2.1.4 Alcance y efectividad del sistema

A partir de este relevamiento, y a causa del problema con respecto a la disponibilidad y visibilidad de los datos ya mencionado, se planteó inicialmente la siguiente estructura como solución:


*Ilustración 4: Esquema de la solución general 1*
(Imagen propia, basada en la práctica)

Como bien se puede apreciar en la ilustración 4, en primer lugar, la sonda multiparamétrica (antes descripta) toma las muestras de los sensores y, mediante el Arduino, enlaza una comunicación inalámbrica con la Raspberry Pi 2b a través de dos módulos LoRa (uno configurado como emisor y otro como receptor, respectivamente). Luego, mediante la implementación de un servicio, se almacenan los datos recibidos a una base de datos alojada

en el servidor productivo de la Universidad Nacional Arturo Jauretche, que se encargará de responder a las peticiones generadas por el sistema multiplataforma desarrollado.

En el presente proyecto, se debe mencionar que para simular el funcionamiento, tanto del entorno de la Raspberry como de la transmisión de datos a través de los módulos LoRa, partimos de la base de que damos por hecho que el Arduino almacena los datos en un archivo Excel y dicha información es consumida por un servicio que se ejecuta en Linux y que realiza las lecturas y actualizaciones correspondientes a la base de datos. Por lo que, teniendo en cuenta estas consideraciones, el siguiente esquema sistematiza todo el proceso:

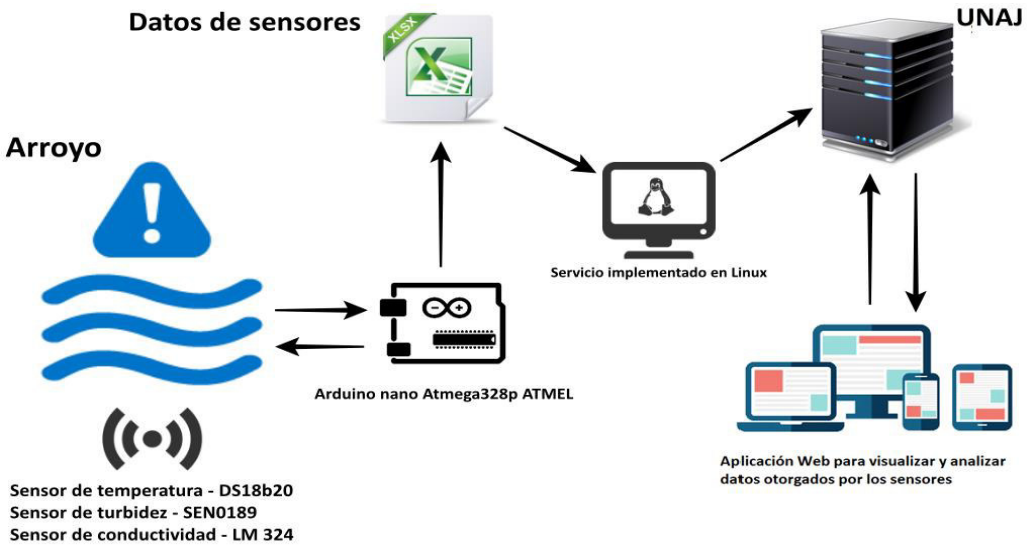



*Ilustración 5: Esquema de la solución general 2*
(Imagen propia, basada en la práctica)

Para concluir con la fase de análisis y relevamiento, podemos decir que ya se describió el comportamiento completo del sistema de sensores por abastecer, como así también ya se revela más claro el alcance final del proyecto.

## 2.2 Definición y diseño de las tecnologías implicadas

En esta fase se desarrolló el diseño de las tecnologías que permiten satisfacer la solución propuesta. Básicamente, se llevaron adelante las siguientes tareas:

- Investigación
- Análisis funcional

- Diseño visual y de interfaz

Cuando se comenzó a planificar este proyecto, la idea de la disponibilidad de datos y la facilidad de su acceso eran primordiales, ya que el procedimiento para acceder a la memoria SD del dispositivo para visualizar la información, como ya se refirió, no era muy óptimo. Por lo que se decidió desarrollar un sistema multiplataforma, que hace referencia a una aplicación que puede ser implementada y que inter-opera en varios dispositivos y plataformas, tanto móviles como de escritorio. La gran diferencia es el código en el que, en cada caso, se escribe.

Antes de describir entonces las tecnologías utilizadas, definiremos y presentaremos el esquema de una aplicación web:

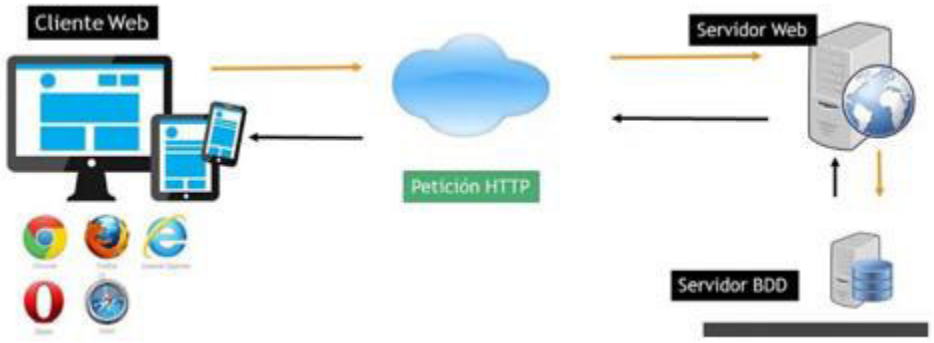



*Ilustración 6: Esquema de una aplicación web*
(Imagen propia, basada en la práctica)

Claramente como ilustra la figura, una aplicación web funciona en base a tres niveles:

- **El del Navegador web (o Cliente Web),** que se encarga de interpretar el lenguaje codificado de la aplicación. Nos referimos a lenguajes informáticos como HTML, CSS o JavaScript.
- **El del Servidor web,** que es el código al que apela el navegador. Puede encontrarse en la nube o en una red local.
- **El de la base de datos (Servidor BDD),** que **es** el lugar de almacenamiento de los datos de la aplicación.

Básicamente, el funcionamiento es el siguiente: el Navegador web realiza peticiones HTTP al Servidor web con el fin de poder acceder a los datos almacenado en la base de datos. En base a esta arquitectura, se decidió la utilización de las siguientes tecnologías:

### 2.2.1 NodeJS

Tal como se menciona en el sitio oficial, es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no se limita a esta), está basado en el lenguaje de programación JavaScript, es asincrónico, tiene I/O de datos en una arquitectura orientada a eventos y basada en el motor V8 de Google. NodeJS fue creado con el objetivo de ser útil en la creación de programas de red, altamente escalables, como, por ejemplo, los Servidores web. NodeJS tiene como gran ventaja la creación de tareas asincrónicas y en segundo plano.

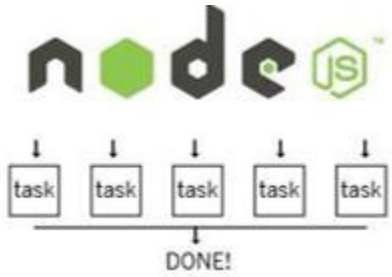



*Ilustración 7: Paralelismo Nodejs*
(Imagen propia, basada en la práctica)

Las ventajas de esta tecnología son:

- Posibilita la asincronía (el no bloqueo)
- Permite NPM (comunidad): la gestión de librerías
- Habilita *Single thread* (el paralelismo)
- Posee librerías propias
- Tiene código abierto
- Es una multiplataforma: se puede ejecutar en cualquier lugar (por ejemplo, en Raspberry Pi)
- Está orientada a Eventos
- No se limita solo a servidores HTTP

Con esta tecnología se desarrollará el servicio encargado de la lectura y el almacenamiento de los valores arrojados por los sensores. Se pensó en Nodejs, ya que, como se mencionó,

por sus características se complementa muy bien con Raspberry, y se especializa en ejecuciones en paralelo y asincrónicas, lo cual resulta muy eficiente para este tipo de tareas.
En cuanto a la conexión con la base de datos, se decidió utiliza el módulo/biblioteca Moongose, que permite definir esquemas y modelos en nuestro código Nodejs con la misma estructura que utilizan nuestros documentos de MongoDB, base de datos que se referirá a continuación.

### 2.2.2 MongoDB

Como se menciona en su presentación oficial, se define como una solución de base de datos NoSQL, es decir, que su estructura y funcionamiento difieren enormemente de una base de datos MySQL, PostgreSQL o MariaDB (relacionales). En contraste, MongoDB es una base de datos muy poderosa construida sobre documentos, los cuales son equivalentes a un objeto de JavaScript, en lugar de las filas y columnas tradicionales de una solución basada en SQL. Obviar el uso del Mapeo Relacional de Objetos u ORM (*Object-Relational Mapping*) en la base de datos permite una más fácil manipulación, transferencia de datos y escalabilidad, ya que puede aumentar la cantidad de datos considerablemente, sin que esto afecte su rendimiento. Además, MongoDB es una solución gratuita y de código abierto que provee muy alto rendimiento y disponibilidad.

A continuación, se visualiza la estructura de Mongodb:

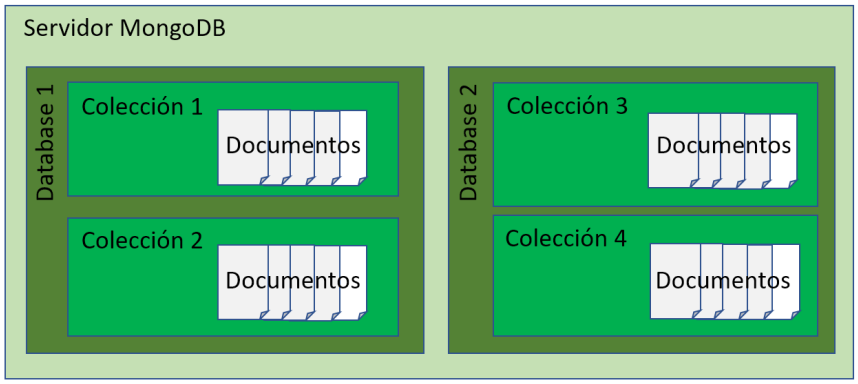



*Ilustración 8: Lima, Emil (2017). Esquema de la base de datos en MongoDB. [Figura]*
(Recuperado de https://lvemil.wordpress.com/2017/06/11/primeros-pasos-con-mongodb-y-net/)

Se optó por esta base de datos ya que, como se mencionó, nos permite escalabilidad y manipulación de grandes cantidades de información. En el caso en que nuestra aplicación, en un momento dado, necesite almacenar mucha cantidad de registros de estados de las aguas de arroyo (lo que implica mucha cantidad de sensores), en ese escenario, es cuando cobra sentido esta decisión tecnológica del diseño propuesto.

### 2.2.3 API REST/Spring Boot

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el *software* de las aplicaciones. API significa "interfaz de programación de aplicaciones". Es la forma por la que muchos sitios web o aplicaciones web obtienen información, mediante peticiones a través de internet, de cierto recurso o *endpoint*. Por ejemplo, en nuestro caso, los datos de cada arroyo. Algunas de las características de las API-REST son:

- Presentan una interfaz uniforme: la interacción entre el cliente y el servidor se realiza de manera uniforme, lo cual simplifica y separa la arquitectura.
- Realizan peticiones sin estado: las peticiones que se realizan no almacenan ningún tipo de información entre ellas, y esto proporciona un mayor rendimiento.
- Posibilitan un almacenamiento en *caché*: esta característica permite guardar en la memoria ciertas respuestas de datos que no cambian en un corto lapso de tiempo.
- Habilitan la separación de cliente y servidor: reducen el acoplamiento entre el aplicativo y el servicio a base de datos.
- Permiten un sistema de "capas": el cliente puede estar conectado mediante la interfaz al servidor o a un intermediario, que para él es irrelevante y desconocido. El uso de "capas" o servidores intermedios puede servir para aumentar la escalabilidad o para implementar políticas de seguridad.

A su vez, las operaciones más importantes que nos permitirán manipular los recursos son cuatro: "get" para consultar y leer, "post" para crear, "put" para editar y "delete" para eliminar. A continuación se muestra una ilustración en donde se puede visualizar el accionar de la API Rest:

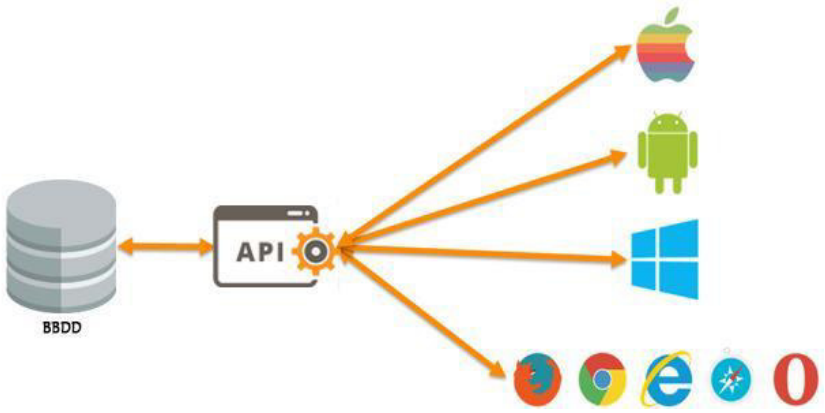



*Ilustración 9: Santamaría, Juan (2016). Esquema de la API. [Diagrama]*
(Recuperado de https://juansm.com/como-crear-una-api-para-tu-aplicacion-web/)

Para desarrollar la API se utilizó Spring Boot. Para comprender esta tecnología, primero, debemos entender cómo trabaja el *framework* Spring de Java. El Spring Framework simplifica el desarrollo de las aplicaciones Java, independientemente de si se trata de aplicaciones web ordinarias o sin conexión web. Sus mayores ventajas son, por un lado, que tiene un código fuente más simplificado y, por otro, una menor dificultad en los ajustes**.** Con el fin de ofrecer al programador dichas comodidades, el *framework* presenta los siguientes principios de funcionamiento:

- **Inyección de dependencias (DI)**: es un patrón externo que regula de antemano las dependencias de los objetos. Para este fin, Spring Framework utiliza los componentes JavaBeans, y estos actúan en Java como un contenedor para la transmisión de datos.
- **Programación orientada a aspectos (AOP)**: posibilita estructurar sintácticamente los aspectos, lo que tiene la ventaja de permitir que el código del programa se separe de los procesos técnicos, como solución de errores, validación o seguridad.
- **Plantillas:** son clases que simplifican el trabajo con API, proporcionando la gestión automática de recursos, el tratamiento uniforme de errores y otras facilidades de asistencia técnica.

A continuación, observamos el esquema completo de un entorno Spring Framework:

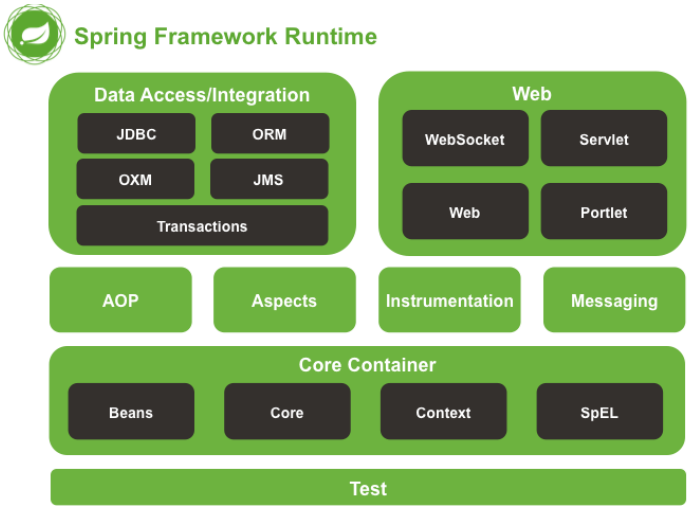



*Ilustración 10: Spring.io, guía para desarrolladores (2014). Esquema de un sistema basado en Spring Framework. [Figura]*
(Recuperado de https://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/overview.html)

En la ilustración 10, podemos distinguir tres secciones importantes:

- **La del Core Container** (núcleo del *framework*), que contiene los módulos básicos, beans (clases Java que permiten encapsular información para estructurar y reutilizar el código fuente) y las expresiones y contextos para poder levantar el entorno de trabajo.
- **La del Data Access/Integration** (acceso e integración de los datos), que consiste en acciones sobre los datos que se van a manejar, es decir, posibilita configurar la base de datos, mapear la información y manipular mensajes y transacciones entre módulos dentro del entorno.
- **La de la Web**, que consta de la configuración y la manipulación de todos los módulos web necesarios.

Una vez definido el concepto general de Spring, continuamos describiendo su derivado: Spring Boot.

Según el sitio oficial, el Spring Boot es una solución para crear aplicaciones basadas en Spring, de una manera más rápida, autónoma, y con características deseables para la

producción. La principal idea detrás de esta tecnología es ofrecer la posibilidad de simplificar la declaración de dependencias de un proyecto y su despliegue en un servidor en particular, permitiendo de esta forma que el desarrollador se enfoque, exclusivamente, en las lógicas/reglas del negocio. Las principales características que podemos destacar son:

- Posee contenedores javas embebidos (por ejemplo, Tomcat).
- Es soporte para los manejadores de dependencias Maven y Gradle.
- No genera código ni configuraciones XML.
- Posee características que facilitan el trabajo en producción de aplicativos.
- Ofrece alta velocidad en la creación de un entorno y proyecto basado en Spring.

En la siguiente figura, se puede observar de forma general cómo está constituido Spring Boot:

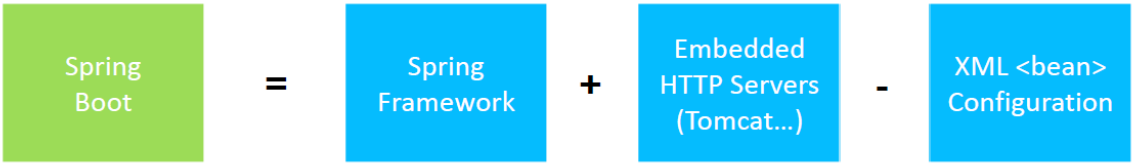



*Ilustración 11: Esquema de un sistema basado en Spring Boot*
(Imagen propia, basada en la práctica)

### 2.2.4 Ionic Framework

Esta tecnología fue desarrollada para generar aplicativos híbridos, que son aplicaciones web que permiten utilizar los recursos y el *hardware* de un dispositivo móvil. Para la creación de estas aplicaciones, Ionic Framework se basa principalmente en los lenguajes de programación HTML, CSS y JavaScript. Antes de continuar describiendo este *framework*, se definirá el concepto de "aplicativo híbrido" y qué diferencias presenta en comparación con aplicaciones nativas (desarrolladas en base al sistema operativo del dispositivo) y sitios web. Una aplicación híbrida es una mezcla entre un aplicativo web, que funcionará en un navegador web, y un aplicativo nativo, que se instala en el dispositivo usando su *hardware* para llevar a cabo sus funcionalidades. La gran ventaja de este tipo de desarrollo es que puede ejecutarse en cualquier dispositivo móvil, independientemente del sistema operativo que utilice, diferenciándose de las nativas, que necesitan ser desarrolladas en un lenguaje apto para el sistema operativo en donde se ejecutarán; por ejemplo, para dispositivos con sistema

operativo Android se deberá desarrollar, nativamente, en lenguaje Java y, para el caso de IOS, en lenguaje Swift.

A continuación, se puede observar una ilustración comparativa entre los distintos tipos de aplicaciones:

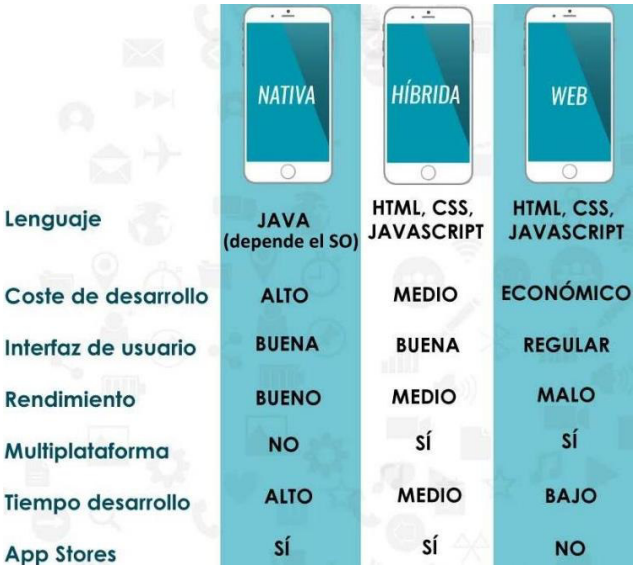



*Ilustración 12: Nubers, guía para desarrolladores (2017). Comparativa de los tipos de aplicativos móviles. [Cuadro comparativo]*
(Recuperada de https://nubeser.com/tipos-desarrollo-apps-moviles/)

**Funcionamiento Ionic**

Esta tecnología incluye otros *frameworks* y librerías para poder funcionar, de las cuales podemos mencionar AngularJs, Apache Cordova, Sass y Nodejs (antes descripto). Como próximo paso, describiremos cada una de estas tecnologías:

- **AngularJs**

  Es una tecnología que se desarrolló en JavaScript puro y es utilizada para desarrollar aplicaciones web dinámicas. La finalidad y orientación de este *framework* es generar aplicaciones SPA (aplicaciones de una página) y, además, otorgar herramientas para desarrollar en la web de forma sencilla. Cuando hablamos de SPA, nos referimos a una web de una sola página en donde la navegación, la carga de datos y las páginas del aplicativo se realizan de forma rápida y realizando llamadas al servidor

asíncronamente, y, aún más importante, sin refrescar el navegador (de forma reactiva).

AngularJs utiliza un Modelo Vista Controlador (MVC), que se comunica con la base de datos, de forma tal que las variables que se encuentran en el controlador se van actualizando y, a su vez, la vista se renderiza. Esta estructura se basa en los siguientes componentes:

- **Módulos:** representan una agrupación lógica, que podríamos llamar "área funcional de nuestra aplicación".
- **Vistas:** hacen referencia a la interfaz gráfica, es lo que visualiza el usuario final.
- **Directivas:** transforman los elementos del DOM para agregar funcionalidad. Cuando se menciona DOM, nos referimos a una estructura lógica de los documentos HTML y XML, que define el modo en que se accede a y se manipulan estos.
- **Servicios:** es la solución para la comunicación y el intercambio de información entre la base de datos y los diferentes componentes declarados en el aplicativo.

- **Apache Cordova**

  Es un *framework* que se encuentra constituido por un conjunto de APIs y que nos proporciona acceso a recursos del dispositivo de forma casi nativa; por ejemplo, el acceso a la cámara, al acelerómetro, a la galería de fotos, entre otros. Esto nos permite poder generar aplicaciones para múltiples plataformas, utilizando el mismo código y lenguaje.

  Las plataformas que soporta son: Amazon Fire OS, Android, BlackBerry 10, Firefox OS, iOS, Ubuntu, Windows Phone, Windows 8, Tizen.

  En el presente proyecto se decidió utilizar Andriod y iOS, básicamente porque son los dos sistemas operativos con más vigencia en el mercado en la actualidad.

- **Sass (*Syntactically Awesome Stylesheets*)**

Es un procesador de hojas de estilos CSS (Hojas de Estilo en Cascada) en tiempo real, es decir, podemos reutilizar propiedades CSS y escribir, de manera rápida, nuestros estilos CSS, y esto da como resultado desarrollos más óptimos.

La necesidad de esta tecnología surge cuando nuestro proyecto va creciendo y las hojas de estilos empiezan a crecer.

En la siguiente ilustración, podemos visualizar cómo es el esquema de Ionic con las tecnologías antes descritas:

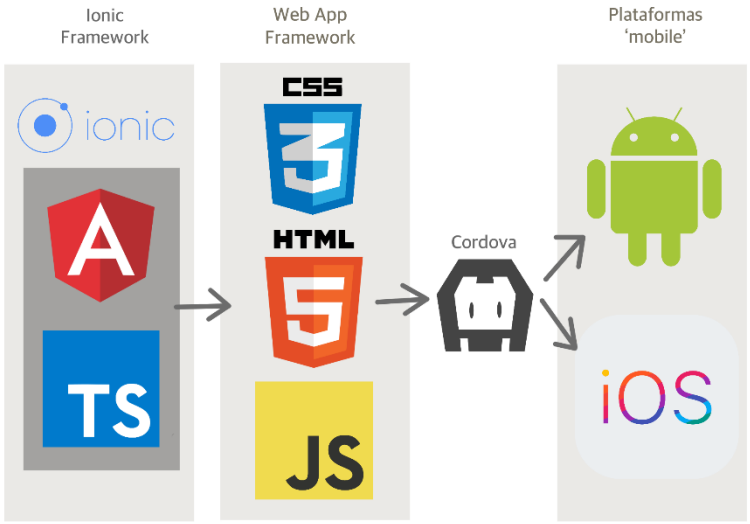



*Ilustración 13: Carvajal, Daniel (2019). Esquema móvil de Ionic. [Diagrama]*
(Recuperada de https://dcarvajal7.es/creando-nuestra-primera-aplicacion-multiplataforma-con-ionic-4/)

Por último, y para concluir con el tema Ionic, se describirá la tecnología utilizada para llevar adelante el aplicativo en versión *desktop*. Para ello utilizaremos el *framework* ElectronJs que, al igual que Apache Cordova, nos ayuda a generar los ejecutables de nuestra aplicación pero, en esta ocasión, para escritorio, ya sea para Windows, Linux o Mac (en el caso del proyecto actual, la desarrollaremos para Windows). ElectronJs utiliza NodeJs para el servidor y el motor de Chromium, para la interfaz gráfica.

### 2.2.5 Docker

Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de "contenedores" de *software*, es decir, nos proporciona un empaquetado de la aplicación con

sus dependencias de forma virtual para poder así ejecutarse en otro servidor. Esto permite una capa adicional de abstracción y virtualización en el nivel del sistema operativo.

Otra ventaja importante que se adquiere al utilizar esta tecnología es que aporta gran flexibilidad y portabilidad al aplicativo.

Utilizaremos esta tecnología, entonces, para poder almacenar en los servidores de la Universidad Nacional Arturo Jauretche el entorno de todo el sistema que plantea el presente proyecto.

Básicamente, se definen archivos "orquestadores" los que posibilitan todas las configuraciones correspondientes para poder disponibilizar todo el ambiente del aplicativo y lograr así funcionalidad en la aplicación web (más adelante, se desarrollará con más detalle la implementación de esta tecnología).

### 2.3 Implementación y funcionalidades del sistema

En esta sección, se describirán en detalle cómo se implementó y qué tipos de funcionalidades posee el sistema. A continuación, se presenta un esquema con las tecnologías ya definidas:

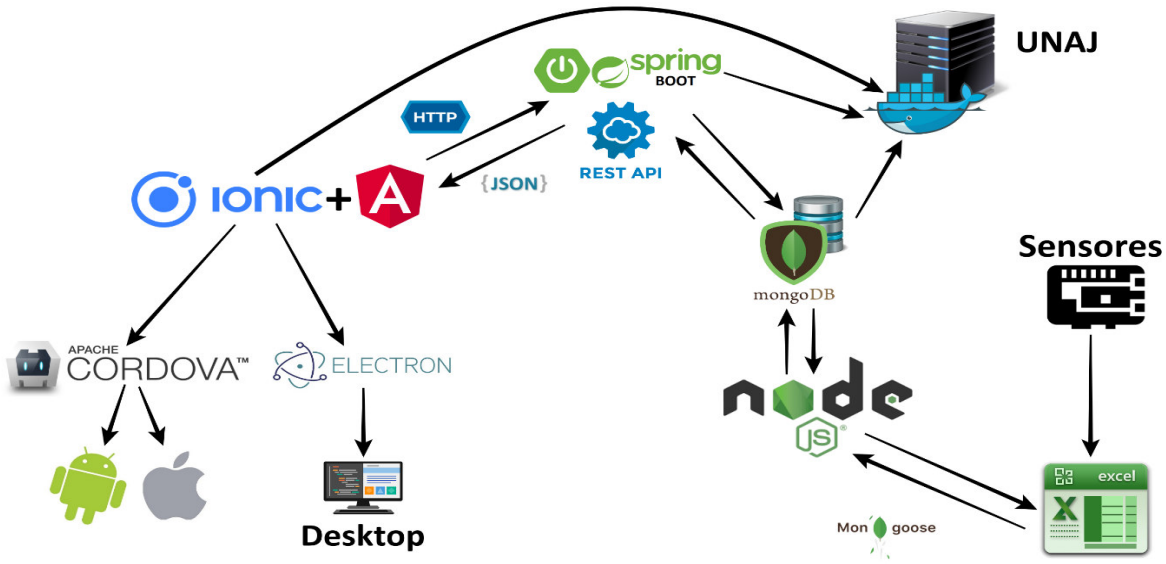



*Ilustración 14: Esquema de las tecnologías usadas*
(Imagen propia, basada en la práctica)

En los siguientes apartados se describirán cada uno de los componentes del sistema completo a nivel funcional y de codificación, incluyendo capturas y explicaciones de comandos/funciones importantes.

### 2.3.1 Linux (servicio de simulación Raspberry)

Tal como se mencionó, este servicio se utiliza para poder realizar la lectura y, posterior, actualización en la base de datos de los valores tomados por el sistema hidráulico. Se configuró para ejecutarse en un sistema operativo Linux, simulando lo que sería un entorno de Raspberry. La lectura se efectúa sobre un archivo Excel, en el cual se encuentra almacenada la información que proveen los sensores (salida de sistema hidráulico) y dos identificadores (uno, para la sonda y el otro, para la localización del arroyo en cuestión). Partiendo de esta información, y con la utilización de NodeJs, se realizaron los siguientes pasos para el desarrollo del servicio:

- Para la lectura de los datos en Excel, se utilizó la librería 'xlsx', la cual nos permite convertir un registro (fila) del archivo en un objeto JSON.
- Una vez generado el JSON, se realiza un formateo de los campos a actualizar.
- Generamos un Excel teniendo en cuenta la fecha de muestra y el identificador de la sonda, logrando así almacenar, de forma más ordenada, los datos muestreados.
- Mediante la utilización de Moongose (mencionado en la sección de tecnologías), se realizó el esquema y modelo a utilizar para poder actualizar, en la base de datos, la información almacenada en el Excel.
- Luego, enlazamos una conexión entre Mongodb y Nodejs para ir actualizando los valores de las sondas de forma asincrónica, utilizando el modelo creado en el ítem anterior.

Como resultado final, el servicio notifica tres mensajes:

- **"**Se actualizaron los valores de los sensores en la base de datos**":** como se indica, se actualizan los valores que posee una sonda dentro del aplicativo mediante el identificador del arroyo.

- "Debe crear localización nueva en la aplicación para poder vincular al nodo con id: xxx": es decir, es necesario dar de alta un "arroyo" dentro del aplicativo (se describe más adelante en el documento).
- "Se agregó nueva sonda (id: xxx) al nodo con id: xxx": como se indica, se generó una sonda nueva dentro del aplicativo.

A continuación, se visualiza la salida del servicio:

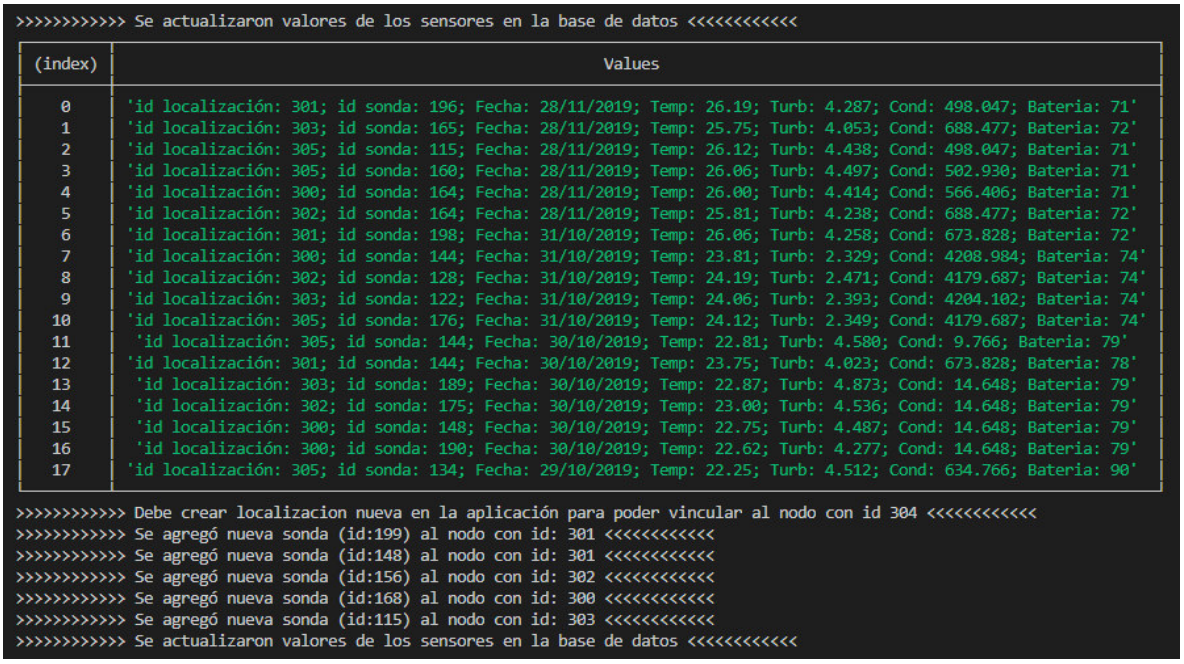



*Ilustración 15: Salida del servicio de Nodejs*
(Imagen propia, basada en la práctica)

Para poder comprender con más claridad el servicio Nodejs, la siguiente ilustración ejemplificada el esquema caracterizado:

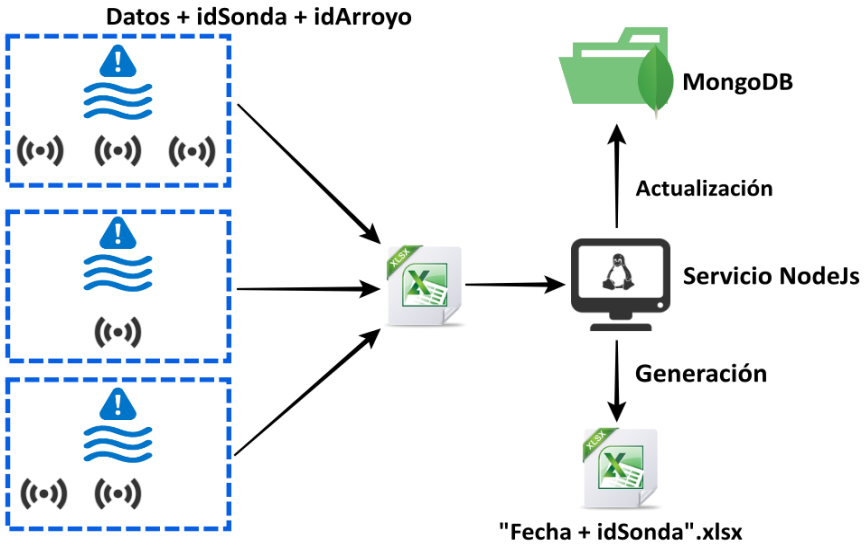



*Ilustración 16: Esquema del servicio de Nodejs*
(Imagen propia, basada en la práctica)

En la ilustración anterior, observamos una situación-ejemplo, en la que hay tres arroyos identificados y la cantidad de sondas que posee cada uno varía. Estas localizaciones, mediante los módulos LoRa (explicadas en el apartado de la solución propuesta), escriben los datos que proporcionan las sondas en un Excel compartido con el servicio NodeJs, y este último se encarga de actualizar la base de datos y, a su vez, generar un registro de la información actualizada, tal como se describió.

Para concluir con este desarrollo, cabe destacar la alta escalabilidad que presenta el servicio planteado, ya que la cantidad de arroyos y sondas no afecta el funcionamiento del servicio desarrollado.

A continuación se presenta un diagrama del flujo que refleja el funcionamiento lógico del servicio:

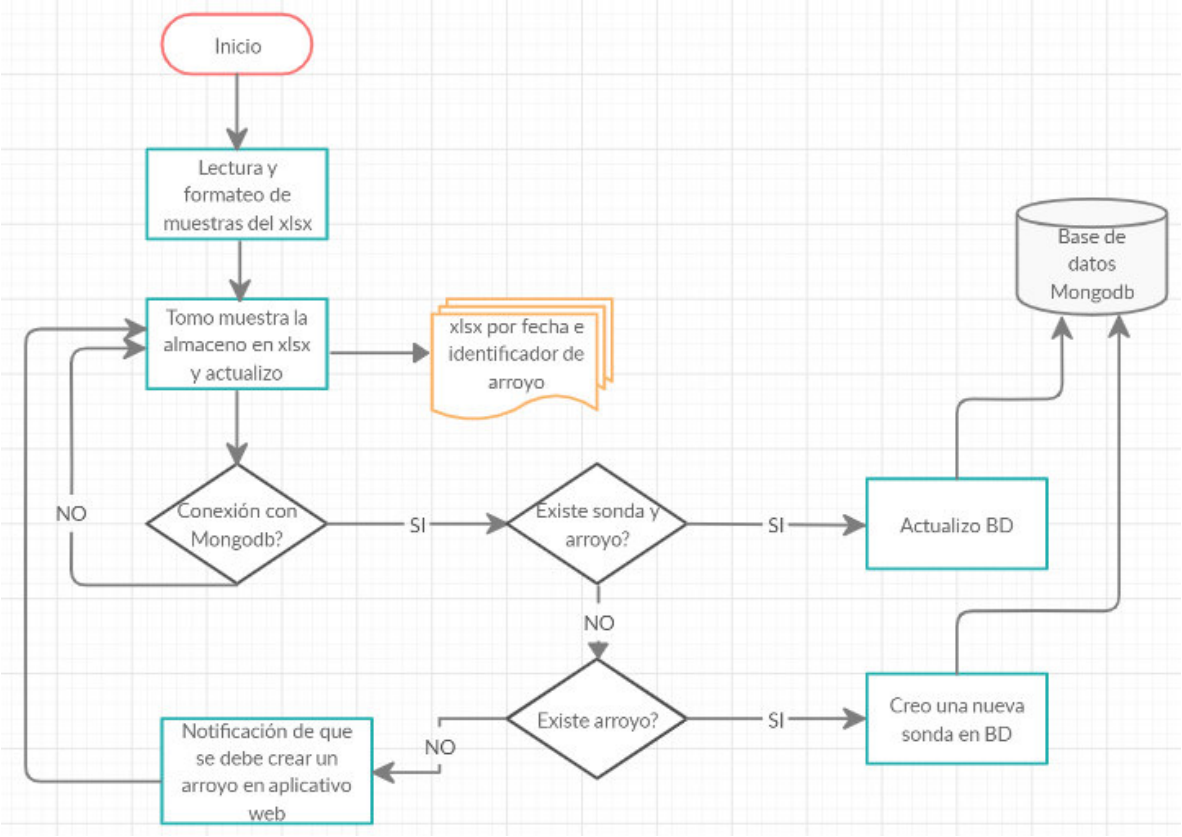



*Ilustración 17: Diagrama de flujo del servicio de Nodejs*
(Imagen propia, basada en la práctica)

### 2.3.2 Aplicación web (para móvil)

En esta sección, se describe el funcionamiento del aplicativo web desarrollado y se incluyen capturas desde un entorno de simulación, propio del *framework* Ionic. Para levantar el entorno de simulación, basta con ejecutar el comando "ionic serve - -lab" en una terminal del sistema donde estemos desarrollando el aplicativo, y esto nos proporcionará la emulación, tanto para iOS como para Android. Antes definimos cómo quedó la estructura de la base de datos MongoDB; a continuación se presentará el modelo de base de datos utilizado.

**Modelo base de datos**

El modelo final con el que interactúa la aplicación web está compuesto por tres colecciones, como se observa a continuación:

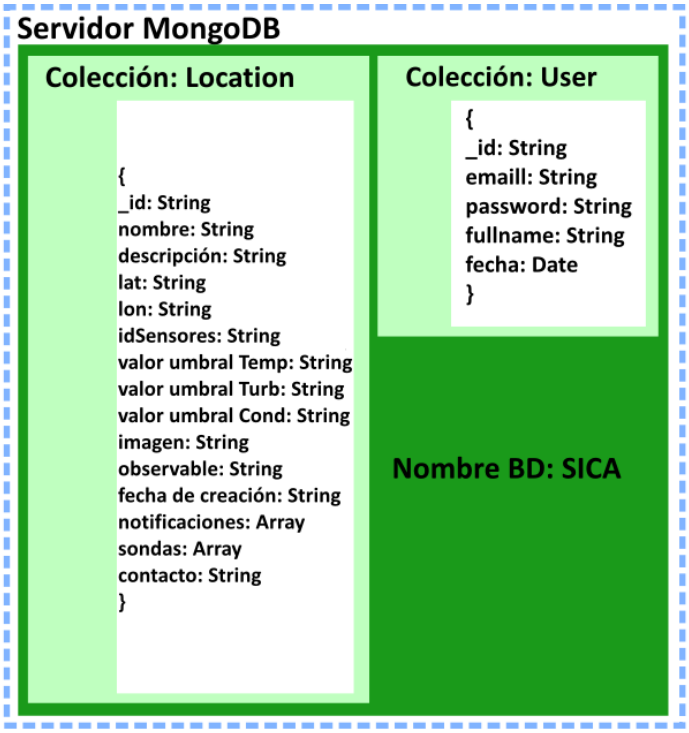



*Ilustración 18: Esquema de la base de datos del aplicativo*
(Imagen propia, basada en la práctica)

A partir de lo expuesto, se presentarán las vistas y las funcionalidades de la aplicación:

**Inicio de sesión**

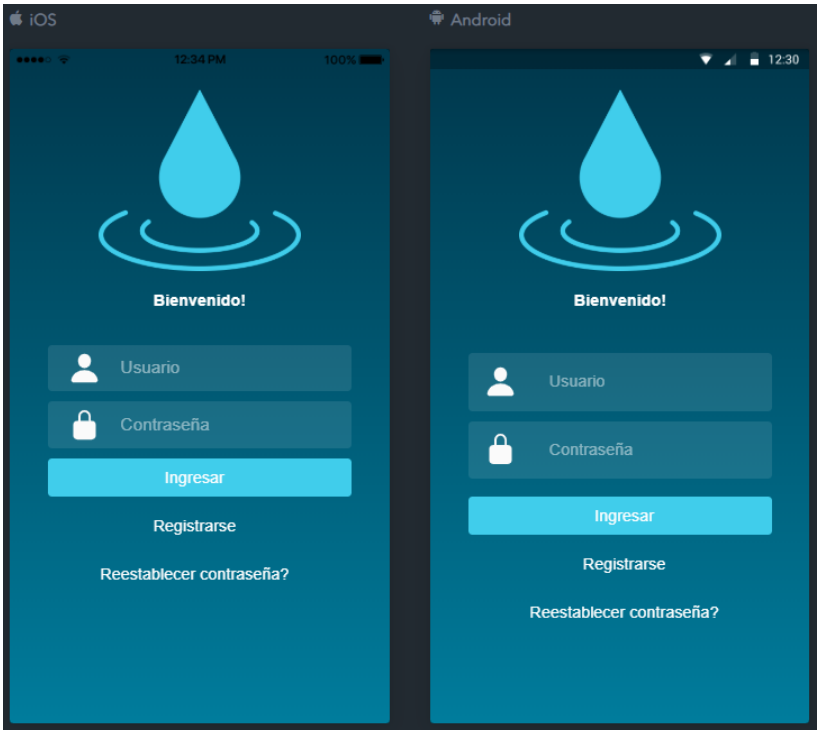



*Ilustración 19: Página del "login"*
(Imagen propia, basada en la práctica)

Cuando ingresamos a la aplicación, nos encontramos con la página de inicio de sesión, que sirve para poder restringir el acceso público a los datos. La contraseña es cifrada (para mejor seguridad) mediante el método de encriptación AES ("encriptación estándar avanzada") que, mediante una clave secreta, definida en la API, encripta y desencripta la contraseña del usuario. Además, es posible reestablecer dicha contraseña, en caso de que se extravíe.

**Registro de usuario**

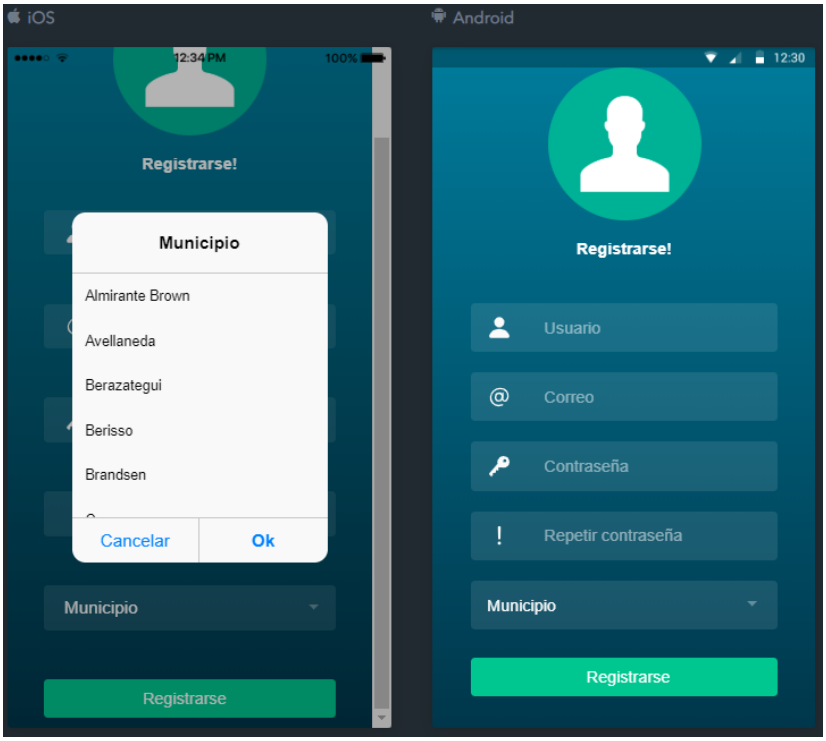



*Ilustración 20: Página del "registro de usuario"*
(Imagen propia, basada en la práctica)

Mediante esta página, tenemos la posibilidad de poder registrar un usuario. Al presionar el botón "registrar", se ejecutan validaciones sobre el formulario, ya sea de formato o mismo si las contraseñas ingresadas no coinciden. Si el registro es exitoso, se genera un mensaje de alerta al usuario y el aplicativo lo redirige a la página de *login* para poder acceder al sistema. Es importante mencionar que, mediante el valor del campo "Municipio", el sistema filtra qué información mostrar al usuario al momento de ingresar al aplicativo, es decir, si el usuario se registra con el valor "Quilmes", solo podrá visualizar información de los arroyos que se encuentren asignados a ese municipio. Esto nos proporciona escalabilidad a la hora de pensar en utilizar el aplicativo para gran cantidad de municipios.

**Notificaciones**

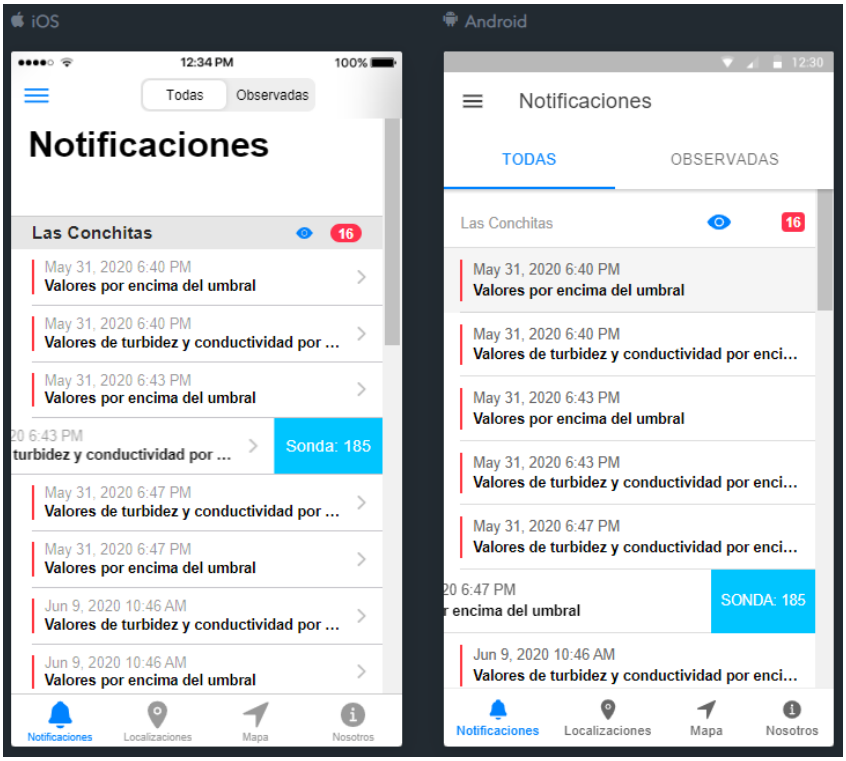



*Ilustración 21: Página de las "notificaciones"*
(Imagen propia, basada en la práctica)

Al ingresar, está la pestaña de "notificaciones", en donde podemos visualizar todas las notificaciones de arroyos vigentes. En la parte inferior de la página, hay cuatro pestañas de navegación que nos dirigen a todas las páginas que posee el aplicativo. Como primera funcionalidad a destacar, en la parte superior, podemos visualizar dos pestañas: "Todas", que hace referencia a todas las notificaciones vigentes en cualquier arroyo, y "Observadas", en donde podremos apartar aquellos arroyos sobre los que se quiere realizar un seguimiento individual y específico, a partir de la información ofrecida en "Notificaciones". Para poder "observar" un arroyo, basta con presionar el ícono a la derecha del nombre de la "localización".

Además, la "notificación" visibiliza la fecha de cuándo fue lanzada y un título descriptivo con respecto al motivo de su lanzamiento. Si deslizamos el ítem hacia la izquierda, es posible visualizar el identificador de la sonda que le corresponde. Las condiciones para que se dé una notificación son:

- Los tres valores actuales de las sonda superan el valor umbral establecido cuando se crea una localización (arroyo).
- El valor de conductividad y turbidez superan el valor umbral establecido.
- El valor de la batería de la sonda es menor al 10%

Por último, cada arroyo posee un contador de notificaciones vigentes para visualizar y actualizar, de forma rápida, la cantidad que posee.

Al presionar alguna notificación, esta nos dirige a su página-detalle, que se describe a continuación:

**Detalle notificación**

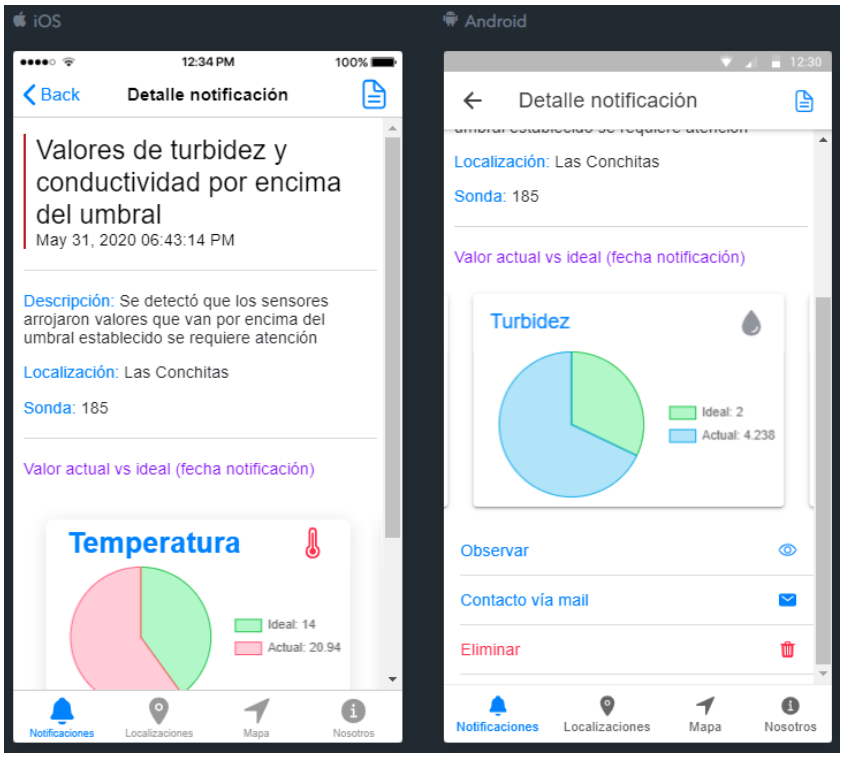



*Ilustración 22: Página del detalle de una "notificación"*
(Imagen propia, basada en la práctica)

Como se puede visualizar, esta página otorga información más precisa de la notificación. Adicionalmente, posee gráficas realizadas mediante la librería ChartsJs de JavaScript, en las que puede observarse una comparación entre el valor actual y el valor umbral al momento de

generarse la notificación. Se configuró un gráfico por sensor y se presenta en forma de *sliders*. Adicionalmente, la página-detalle de la notificación cuenta con tres botones al final de la página, que se utilizan para observar el arroyo, para poder entablar una conversación vía mail (la aplicación utiliza el servicio de correo por defecto) y para eliminar la notificación respectiva.

**Localizaciones**

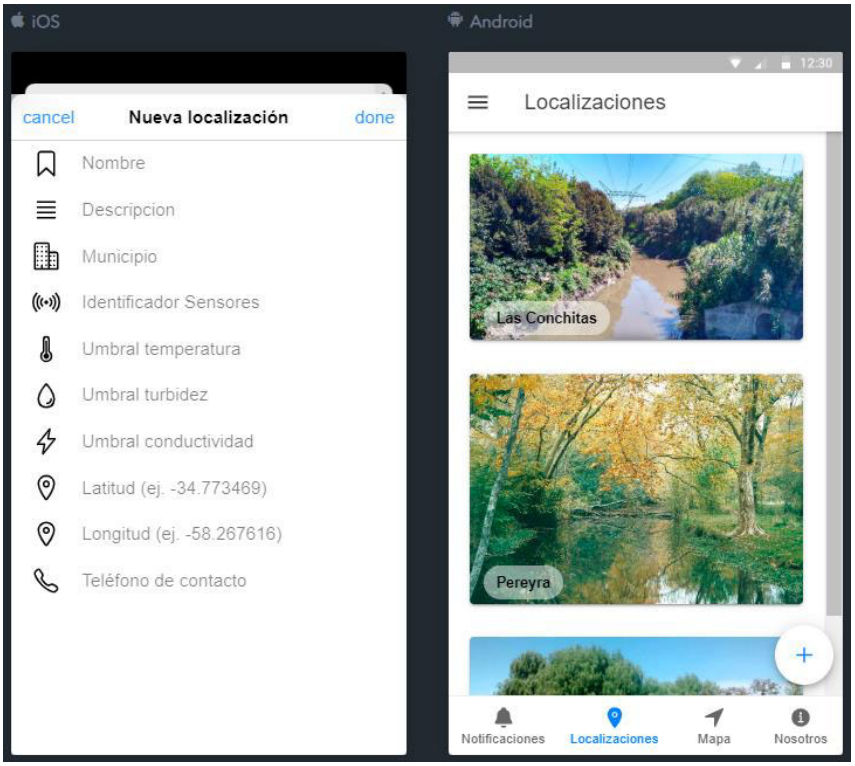



*Ilustración 23: Página de las "localizaciones"*
(Imagen propia, basada en la práctica)

En esta sección, podremos visualizar todos los arroyos vigentes almacenados en la base de datos y, como se refirió, presionando sobre el ítem de cada localización, podremos acceder a su página detalle.

Adicionalmente, en la página de las "localizaciones", hay un botón flotante, en la parte inferior derecha, que sirve para poder acceder a dar de alta una nueva localización (arroyo). Al presionar el botón, se dispara una página secundaria en la que podremos ingresar los datos que correspondan. Lo importante de esta funcionalidad es que, mediante el campo

"Identificador Sensores", se realiza la relación con el identificador del arroyo establecido en el servicio de NodeJs, antes descrito, y esta mención es muy importante, ya que mediante esta relación se produce el enlace entre el servicio y las sondas a actualizar.

**Detalle Localización**

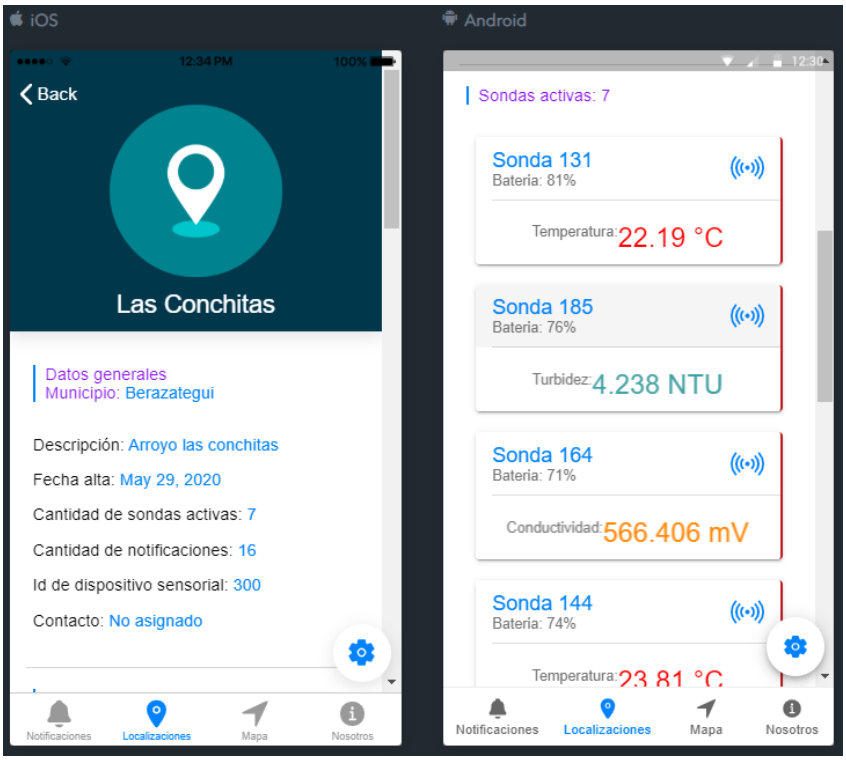



*Ilustración 24: Página del detalle de la "localización" 1*
(Imagen propia, basada en la práctica)

En esta página, logramos visualizar, con mucho más detalle, los datos del arroyo. En primer lugar, se detallan los campos descriptivos sobre el arroyo y un listado con sus sondas vigentes. De forma rápida, este listado nos provee en cada ítem información básica de la sonda, ya sea sobre el estado de la batería como sobre los valores actuales de los sensores (en forma de *sliders* y basándose en la última fecha de la muestra actualizada). Al seleccionar alguno de los ítems, accedemos al detalle de esa sonda.

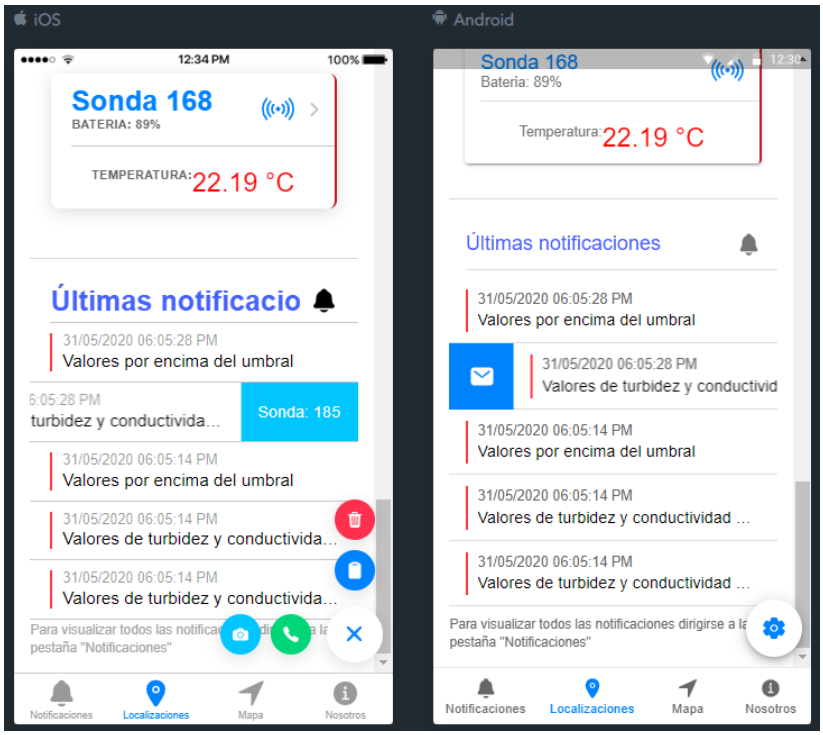



*Ilustración 25: Página del detalle de la "localización" 2*
(Imagen propia, basada en la práctica)

Luego, en la ilustración 25, también se puede observar que se configuró un botón flotante con múltiples funcionales:

- **Actualización de datos**: básicamente, se activa una página secundaria en la que se presenta un formulario para poder actualizar el dato que se necesite (el formulario es idéntico al del alta).
- **Eliminación de datos**: se puede eliminar el arroyo actual y redirigir la página al listado de localizaciones.
- **Establecimiento de Contacto**: permite abrir el servicio de correo vigente en el dispositivo o llamar al contacto ingresado al momento de dar el alta a la localización.

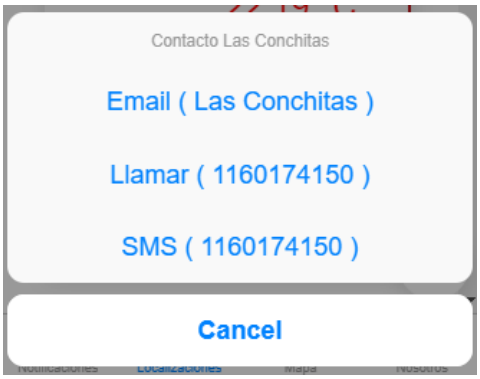



*Ilustración 26: Menú de detalle de la acción "contacto"*
(Imagen propia, basada en la práctica)

- **Acceso a la Cámara**: permite el acceso a la Galería para cambiar la imagen de portada del arroyo o bien se puede utilizar la cámara para poder capturar una imagen del perfil del arroyo en tiempo real.

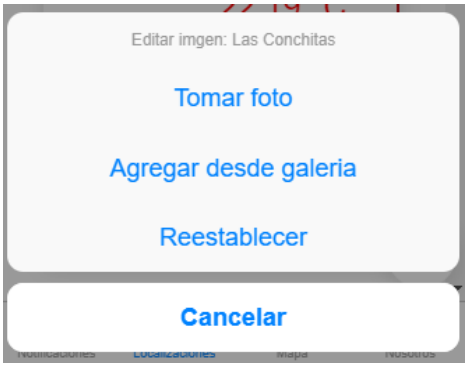



*Ilustración 27: Menú del detalle de la acción "cámara"*
(Imagen propia, basada en la práctica)

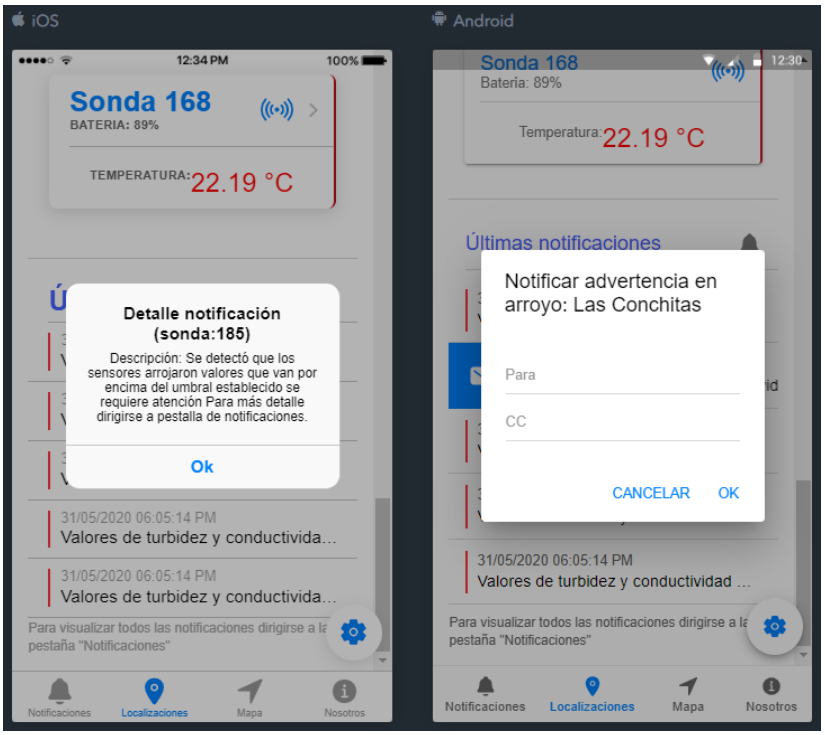



*Ilustración 28: Página del detalle de la "localización" 3*
(Imagen propia, basada en la práctica)

Por último, en la ilustración 28, se describe una sección en donde se visualizan las últimas cinco notificaciones a modo informativo. Al presionar cada ítem, se despliega un detalle más completo y, si deslizamos hacia la derecha, existe la posibilidad de enviar un correo con el dato de los valores actuales que se registraron al momento de lanzarse la notificación.

## Detalle sonda (Android)

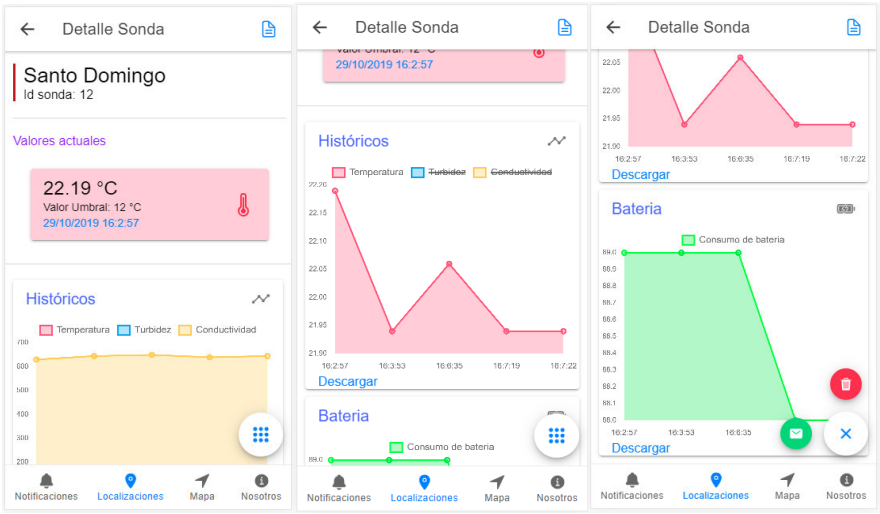



*Ilustración 29: Página del detalle de la sonda Android*
(Imagen propia, basada en la práctica)

## Detalle sonda (iOS)

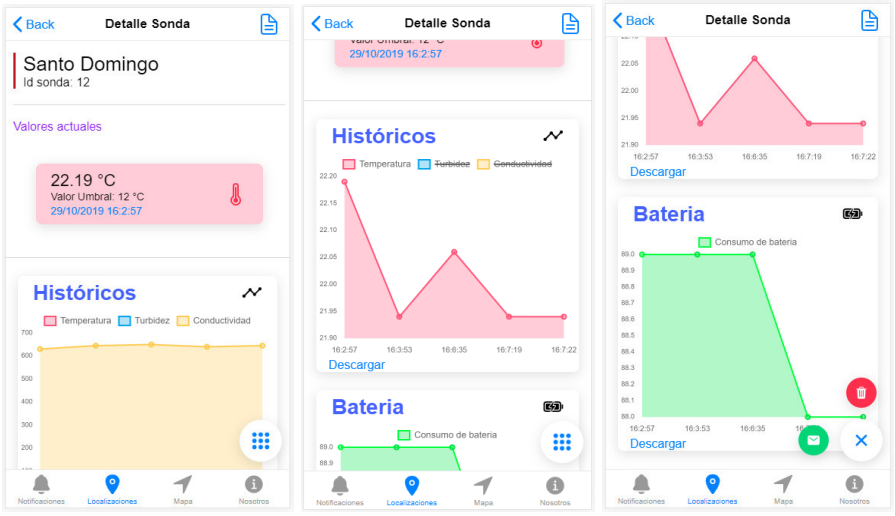



*Ilustración 30: Página del detalle de la sonda iOS*
(Imagen propia, basada en la práctica)

En esta página, observamos tres secciones principales, que informan respectivamente:

- **Los valores actuales**: en forma de *sliders*, muestra los valores de la última muestra actualizada en la base de datos, ya sea de los sensores como del estado de la batería.

- **Un gráfico histórico**: utilizando ChartJs, muestra las últimas cinco mediciones de los sensores pertenecientes a la sonda actual (y permite, de forma dinámica, cambiar de sensor).
- **Un gráfico sobre el consumo de la batería**: utilizando ChartJs, muestra las últimas cinco mediciones del estado de la batería de la sonda actual.

Por último, al igual que todas las páginas de detalles, mediante un botón flotante, es posible eliminar la sonda (y redirige a la página anterior) y conectarse al servicio de correo del dispositivo.

**Sistemas activos (página del "mapa")**

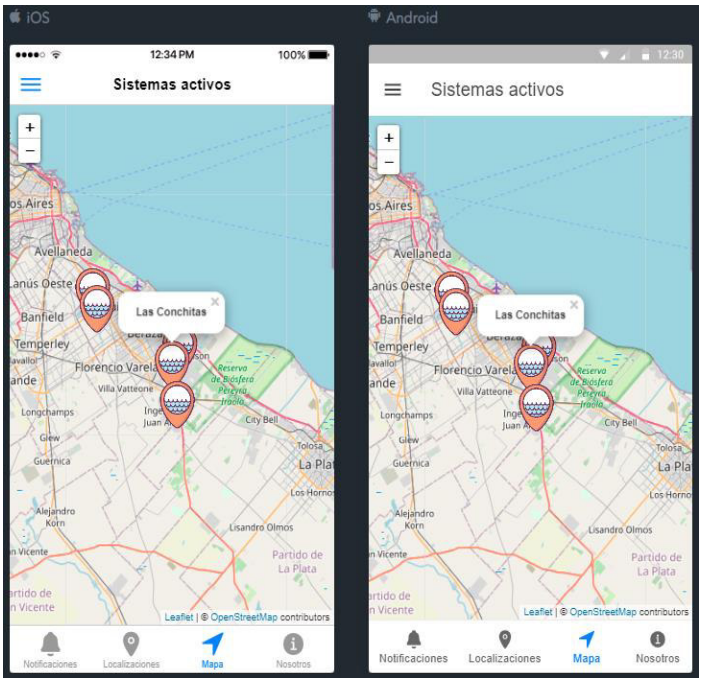



*Ilustración 31: Página del "mapa"*
(Imagen propia, basada en la práctica)

En la pestaña "mapa", podemos visualizar la ubicación real del arroyo. Al momento de dar el alta, se debe ingresar la latitud y la longitud del arroyo, y el sistema se encarga de generar una marca en el mapa en dicha posición.

Se utilizaron los mapas *open source* de OpenStreetMap, debido a que es un sitio gratuito y de fácil implementación. Para dibujar el mapa se utilizó la librería Leaflet de JavaScript.

**Página de contacto**

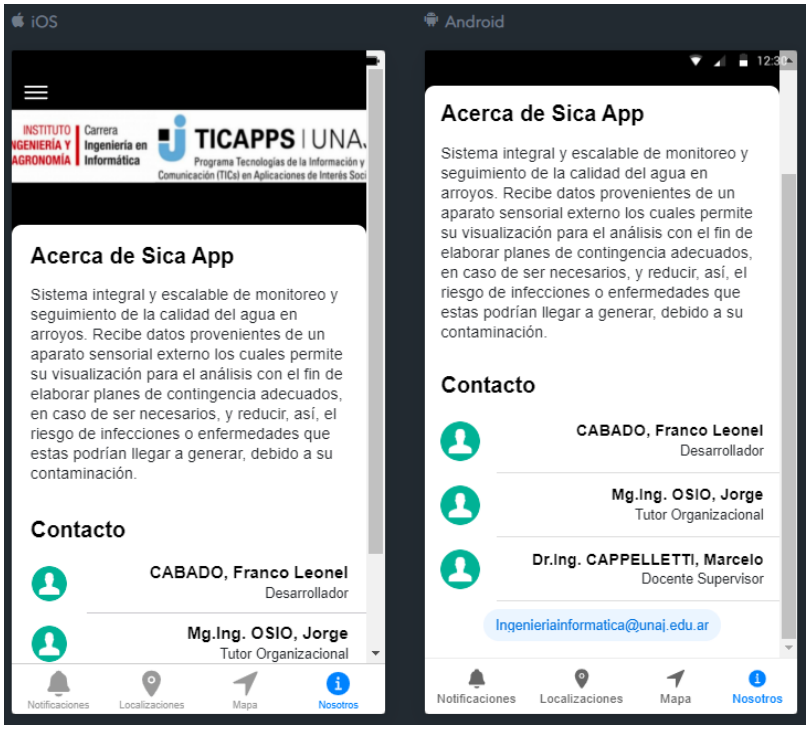



*Ilustración 32: Página del "contacto"*
(Imagen propia, basada en la práctica)

En la pestaña "nosotros", se visualiza una breve descripción del aplicativo y se hace mención a los involucrados en el desarrollo de la presente Práctica Profesional Supervisada.

**Menú desplegable**

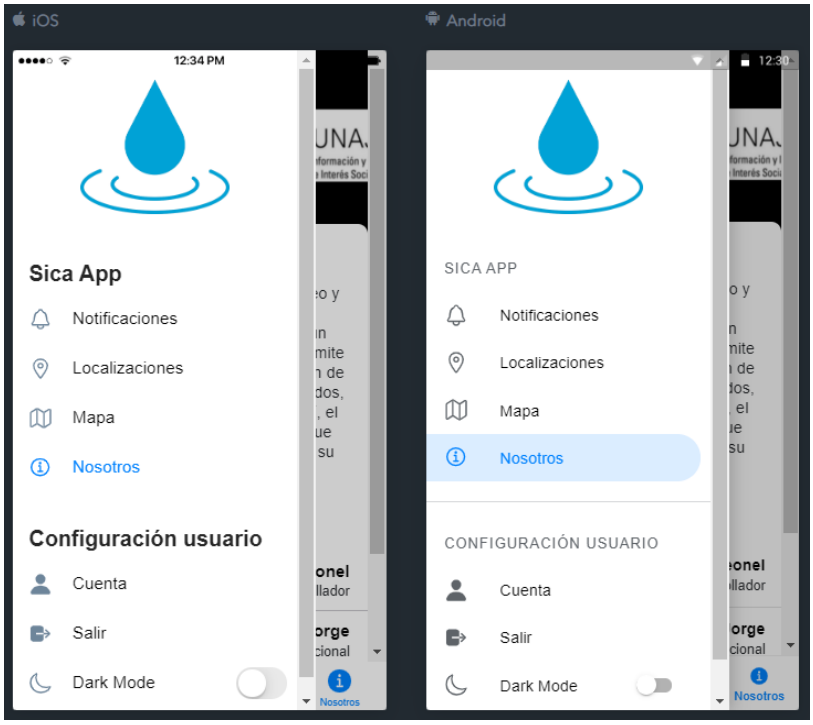



*Ilustración 33: Página del "menú desplegable"*
(Imagen propia, basada en la práctica)

El menú desplegable de la ilustración 33 se utiliza para mejorar la navegación del aplicativo y posee las mismas opciones que las pestañas mencionadas al inicio de este apartado. Adicionalmente, presenta una sección de "Configuración de usuario" en donde podremos realizar tres acciones:

- Revisar la cuenta: presenta un breve detalle del usuario actual y permite editar su información.

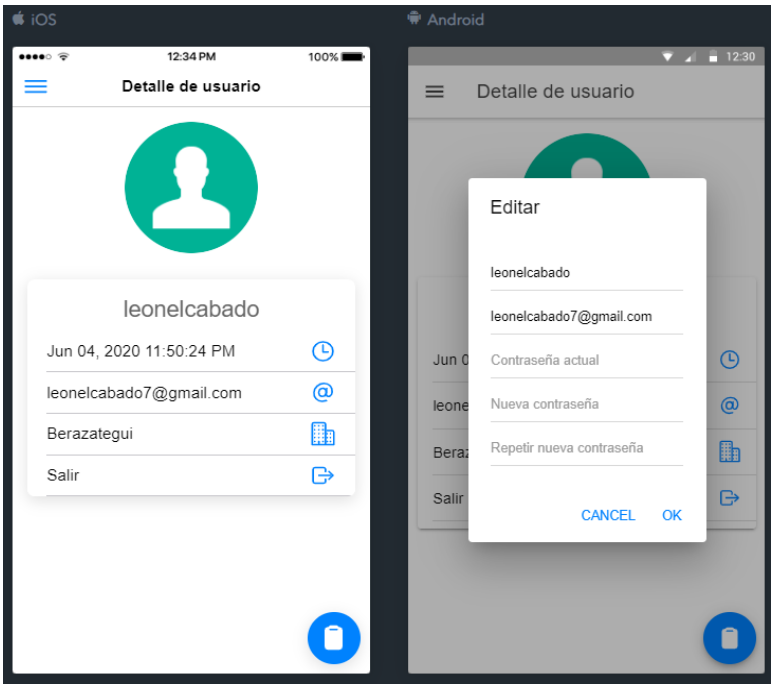



*Ilustración 34: Página de la "cuenta"*
(Imagen propia, basada en la práctica)

- Salir de la cuenta: permite volver al *login* del aplicativo.
- Activar/desactivar el modo de visualización oscuro: mediante técnicas de Sass y Css, se altera el color de toda la aplicación para reducir el riesgo ocular del usuario.

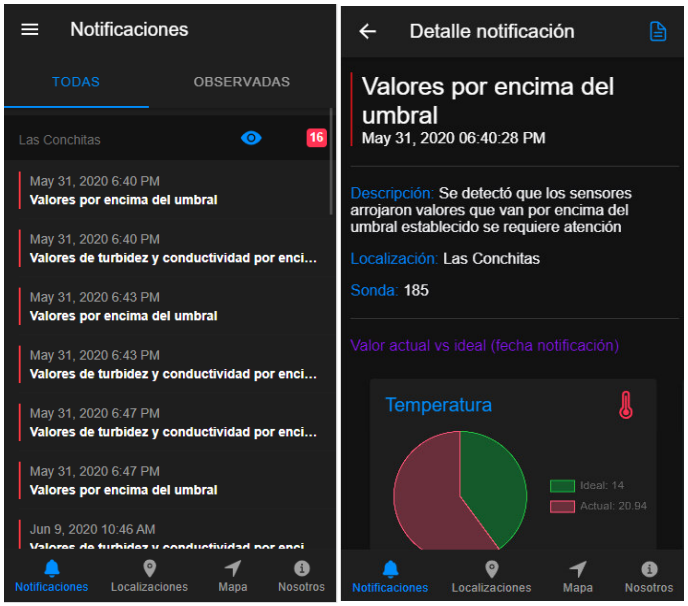



*Ilustración 35: Dark mode*
(Imagen propia, basada en la práctica)

### 2.3.3 Aplicación web (*desktop*)

En este apartado, se describirán los pasos desarrollados para implementar el aplicativo en una computadora de escritorio con sistema operativo Windows. Como bien se mencionó anteriormente, la idea es desarrollar un único código, ya sea para Android, iOS y, como en este caso, Windows. En relación con esto, y mediante algunos comandos y la utilización de las tecnologías (ya mencionadas), existe la posibilidad de generar la aplicación de escritorio de manera rápida y de fácil implementación. Los pasos a seguir son los siguientes:

- **Instalar el Electron JS:**

  Esta instalación debe realizarse mediante el comando "npm install ngx-electron electron" en donde "npm" corresponde al manejador de dependencias de NodeJS. Una vez finalizada dicha instalación, se procede a instalar el módulo "electron-packager" (que, a su vez, será utilizado más adelante para generar el ejecutable final) con el siguiente comando: "npm install electron-packager --save-dev".

- **Integrar al proyecto el módulo Capacitor:**

  Este módulo trabaja de forma similar a Cordova, sería como un intermediario entre los *plugins* de *hardware* y nuestra aplicación. Para poder realizar la integración con Ionic, basta con ejecutar el siguiente comando: "ionic integrations enable capacitor".

- **Modificaciones sobre el código del aplicativo:**

  Si bien la mayor parte del código Ionic será reutilizada por Electrón, hay que tener en cuenta que algunos *plugins*, por ejemplo el de la cámara o el de las llamadas y SMS, no podrán reutilizarse, ya que el sistema operativo Windows no los reconoce. Es, por esto, que existe la necesidad de suprimir algunas funcionalidades (no relevantes) del aplicativo final. Adicionalmente, se realizaron cambios en el código que define la apariencia visual de la aplicación.

- **Integrar y combinar el ElectrónJs al proyecto:**

  Para finalizar, ejecutamos el comando "npx cap add electron", en donde "npx" es un derivado de npm y "cap" corresponde a Capacitor. Esto genera la estructura del ElectrónJs en una carpeta dentro del proyecto. Luego, para combinarlo con el código de Ionic, se procede con "ionic build && npx cap copy": básicamente lo que hace

este comando es, en primer lugar, construir la aplicación Ionic y luego, en segundo lugar, combinarla con ElectrónJs.

Al ejecutar "npx cap open electron" se obtuvieron los siguientes resultados:

**Inicio de sesión**

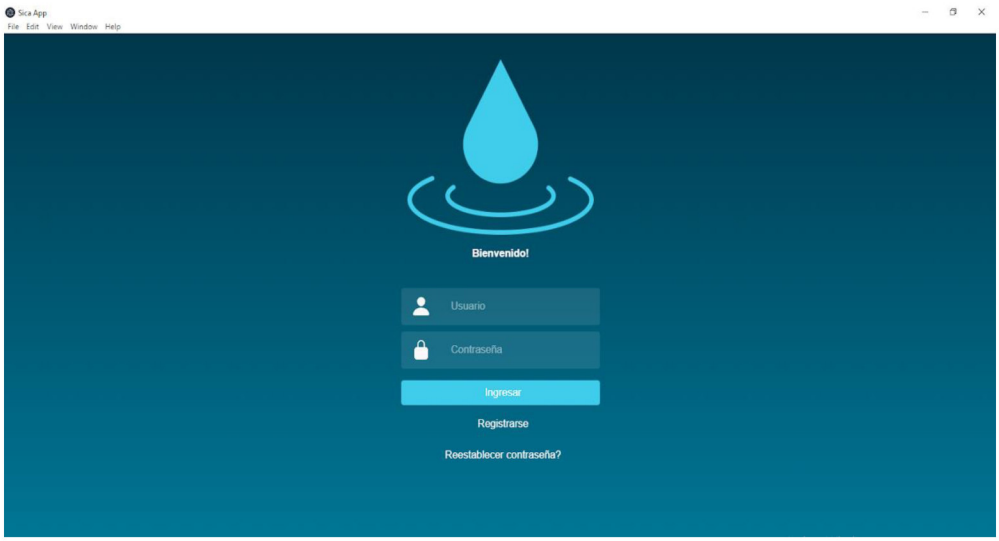



*Ilustración 36: Página del "login" para desktop*
(Imagen propia, basada en la práctica)

**Notificaciones**

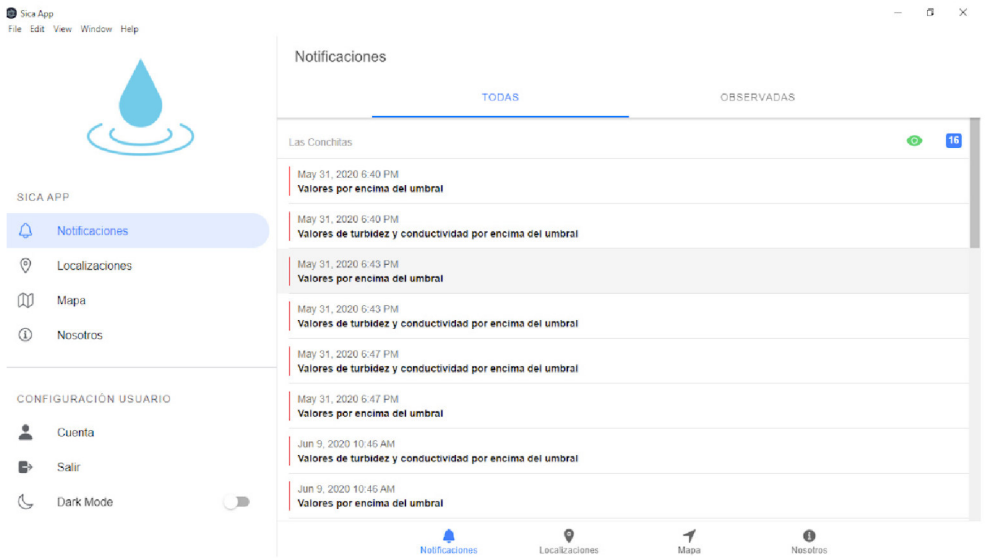



*Ilustración 37: Página de las "notificaciones" para desktop*
(Imagen propia, basada en la práctica)

## Localizaciones

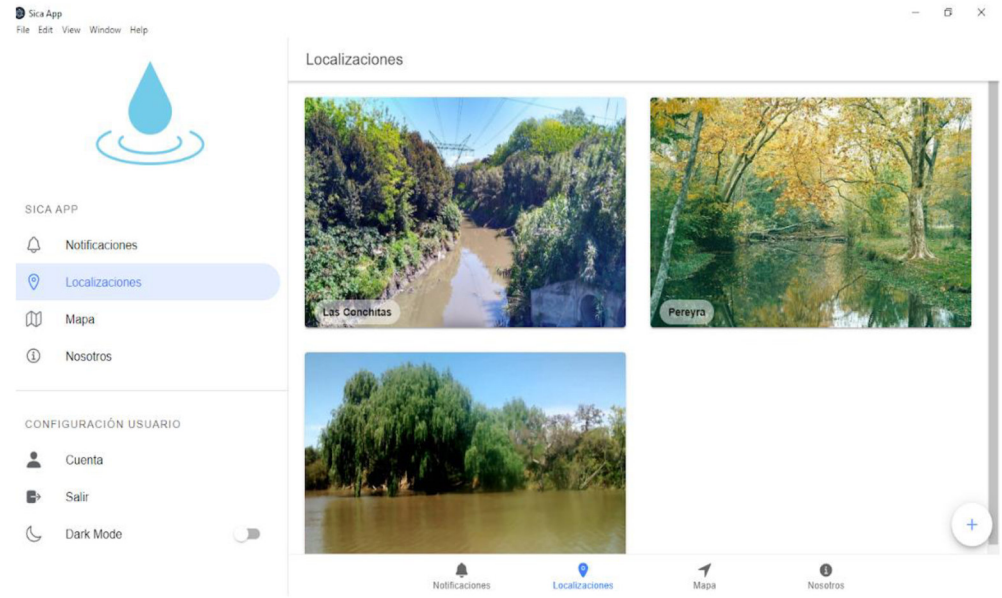



*Ilustración 38: Página de las "localizaciones" para desktop*
(Imagen propia, basada en la práctica)

## Detalle localización

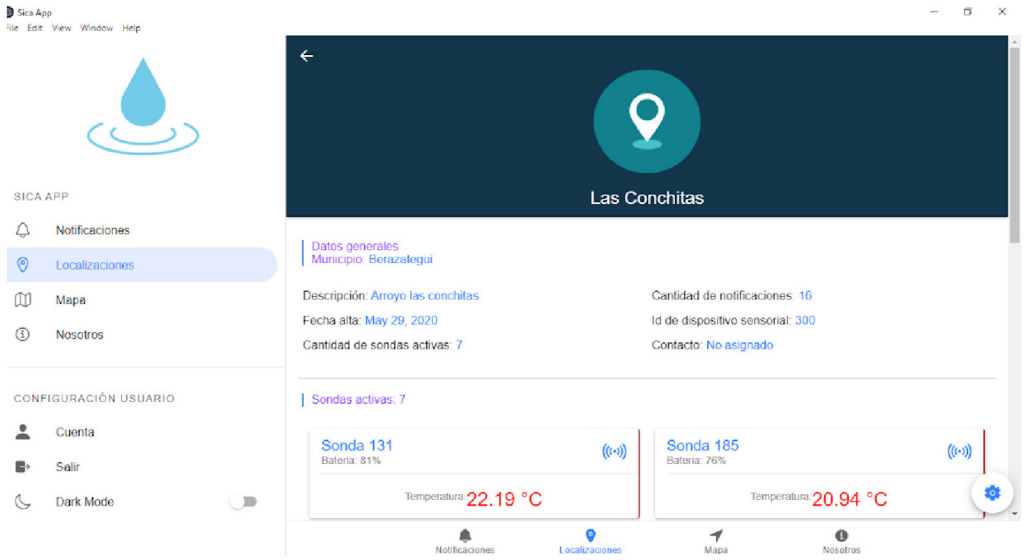



*Ilustración 39: Página del detalle de una "localización" para desktop*
(Imagen propia, basada en la práctica)

## Detalle sonda

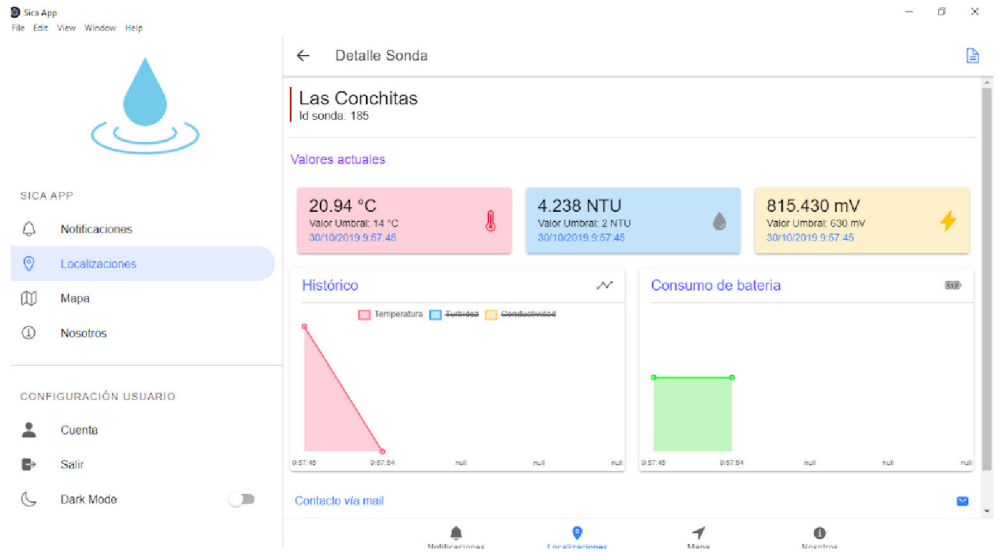



*Ilustración 40: Página del detalle de una "sonda" para desktop*
(Imagen propia, basada en la práctica)

## Detalle notificación


*Ilustración 41: Página del detalle de una "notificación" para desktop*
(Imagen propia, basada en la práctica)

## Mapa

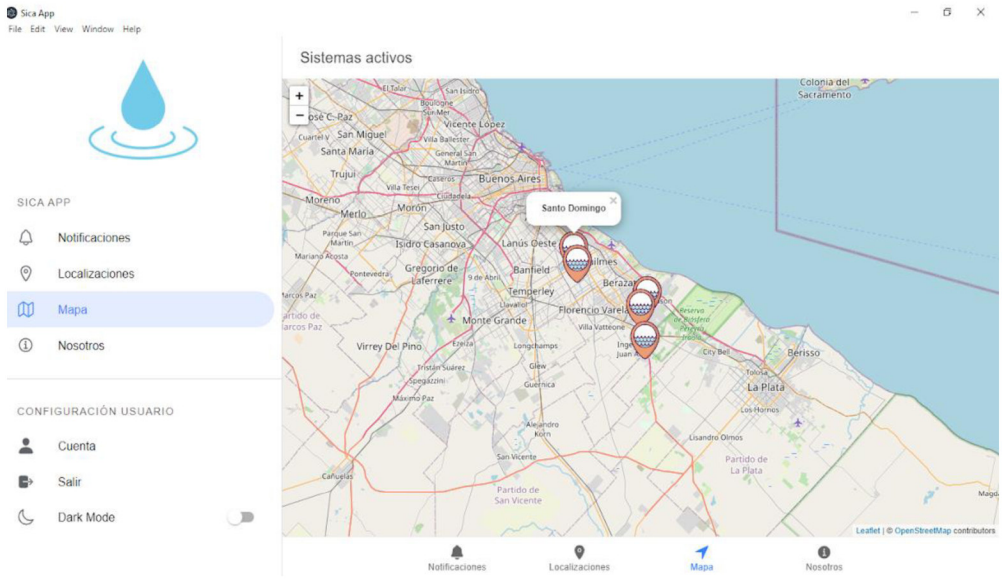



*Ilustración 42: Página del "mapa" para desktop*
(Imagen propia, basada en la práctica)

## Nosotros


*Ilustración 43: Página del "contacto" para desktop*
(Imagen propia, basada en la práctica)

### 2.3.4 Implementación de los aplicativos

En esta sección, se describe el proceso para ejecutar la aplicación en cada sistema operativo, ya sea en un entorno móvil o de escritorio, como se distingue a continuación:

- **Ejecución en entorno móvil**

Para poder instalar la aplicación en un dispositivo Android o iOS basta con utilizar el comando "Ionic Cordova run android/ios --prod", que posee la consola interna de Ionic, sobre el directorio raíz. El funcionamiento de este comando consiste en instalar en el dispositivo el código fuente de nuestro aplicativo, generando el ejecutable de Android con extensión ".apk" y/o el de iOS con extensión ".ipa". A su vez, el comando adicional "--prod" nos proporciona un testeo a nivel productivo de todo el código fuente con la finalidad de detectar cualquier tipo de error que pueda afectar el funcionamiento de la aplicación, antes de generar los ejecutables mencionados.

Si se desea generar el ejecutable de forma independiente de Ionic, es posible utilizar el entorno de desarrollo nativo Andorid Studio o Xcode, respectivamente.

A continuación se visualizan los resultados obtenidos mediante capturas de pantallas en un dispositivo Motorola G6.

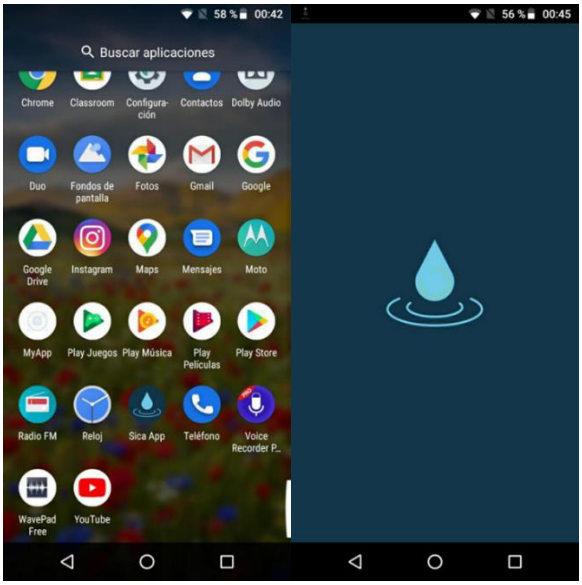



*Ilustración 44: Ejecución de la aplicación en un dispositivo móvil*
(Imagen propia, basada en la práctica)

Se puede visualizar, en primera instancia, el aplicativo ya instalado dentro del dispositivo con el ícono representativo del sistema y bajo el nombre "Sica App". Una vez que presionamos el ícono, accedemos a una página de "bienvenida" y, luego, a la de "inicio de sesión" (antes descripta).

- **Ejecución en entorno de *desktop***

En cuanto al aplicativo de escritorio, se utilizó el comando "electron-packager <directorio fuente> SicaApp --platform=win32". Al ejecutar este comando sobre el directorio raíz del código fuente, se genera una carpeta que contiene empaquetado el instalador del aplicativo y todos sus componentes/librerías de las que depende para su funcionamiento. En nuestro caso, se realizó el instalador en Windows, como se ve a continuación:

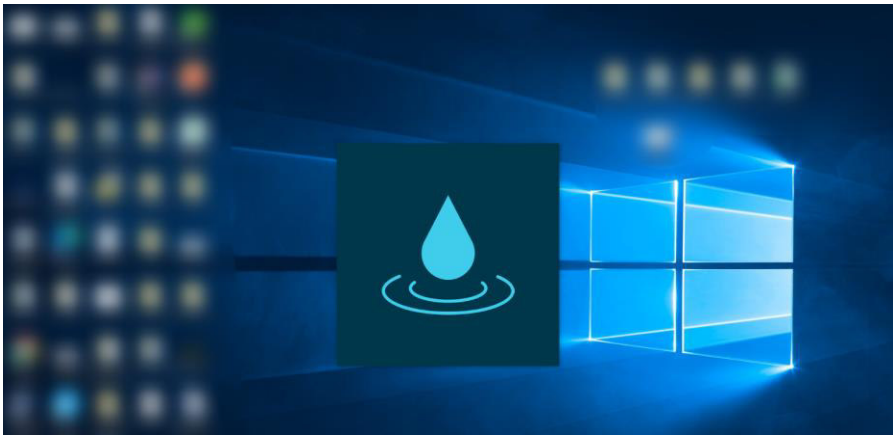


*Ilustración 45: Ejecución de la aplicación en un dispositivo de desktop 1*
(Imagen propia, basada en la práctica)

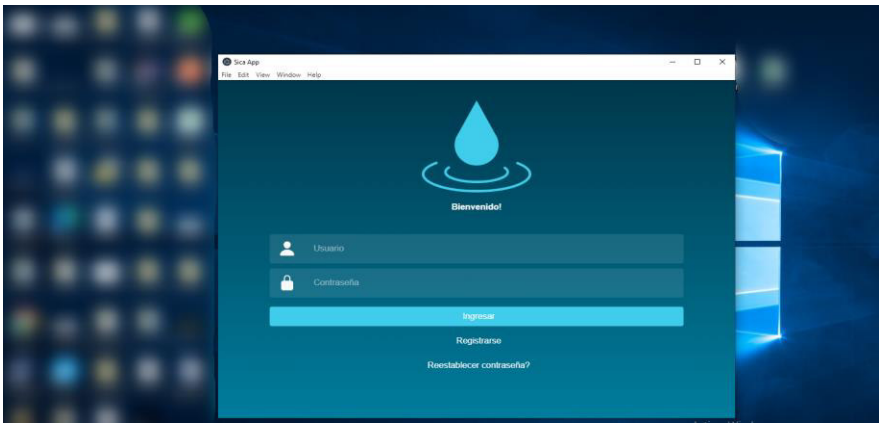



*Ilustración 46: Ejecución de la aplicación en un dispositivo de desktop 2*
(Imagen propia, basada en la práctica)

### 2.3.5 Implementación en el servidor productivo (utilizando Docker)

Luego de un relevamiento con el área de sistemas de la Universidad Nacional Arturo Jauretche, se determinó usar un *cluster* de servidores (un conjunto de servidores que se construyen e instalan para trabajar como si fuese uno solo) para iniciar los distintos servicios que se utilizan en la universidad. El *cluster* se conforma de archivos de configuración que manipulan los distintos contenedores Docker (mencionado en el apartado 2.2.5 del presente informe) y estos ficheros poseen el nombre de "docker-composer.yml". La composición de estos archivos se basa en comandos que se utilizan para ubicar y ejecutar los contenedores del *software*.

A continuación, se puede visualizar una ilustración que detalla el flujo de la implementación del Docker desarrollado:

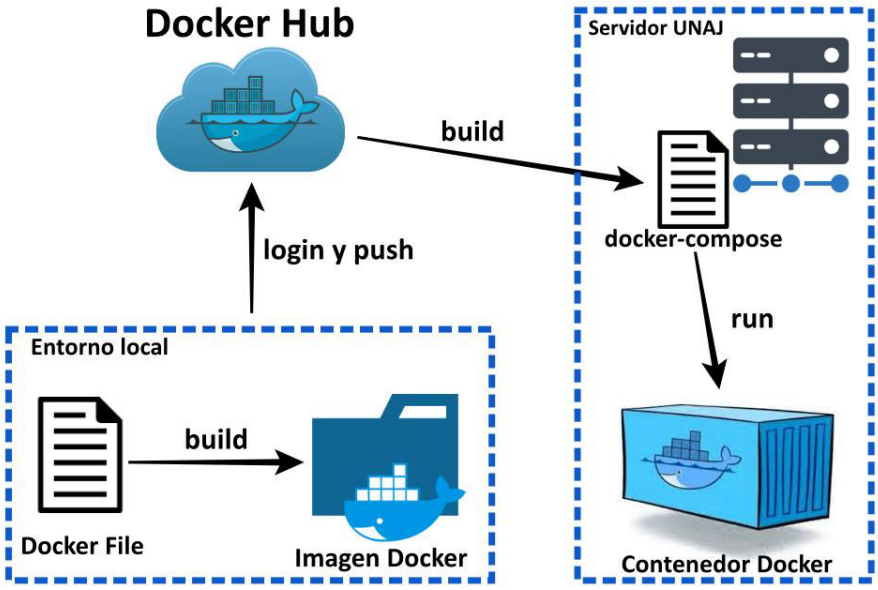



*Ilustración 47: Esquema del Docker desarrollado*
(Imagen propia, basada en la práctica)

En primer lugar, se desarrollaron los "Docker File", que consisten en una serie de comandos que se utilizan para construir una imagen Docker (un archivo informático donde se almacena una copia o "imagen" exacta de un sistema de archivos de una tecnología y bajo la configuración de Docker). Las tecnologías necesarias, a ejecutar en el servidor de la universidad, son: la Api Rest, la base de datos MongoDB y el entorno de Ionic (por si en un futuro fuera necesario realizar una versión de sitio web del aplicativo).

A continuación procederemos a describir cada Docker File desarrollado:

- **Ionic parte front-end**

Se utilizó la imagen agileek/ionic-framework que se encuentra en DockerHub (repositorio público que está en la nube y se utiliza para distribuir contenidos referentes a Docker) y que nos proporciona un ambiente muy completo, basado en una sistema operativo Ubuntu. Como se puede observar en la ilustración 48, en primer instancia, copiamos el código fuente de la aplicación desarrollada a la carpeta "/www/app" dentro de la imagen y luego, en una segunda instancia, exponemos los puertos que utilizaremos e inicializamos el servidor de Ionic.

Cliente_SicaApp > Sica-Front > Dockerfile > ...

```
1   FROM  agileek/ionic-framework
2   LABEL maintainer="leonelcabado"
3
4   COPY . /www/app
5
6   WORKDIR /www/app
7
8   EXPOSE 8100 35729
9
10  ENTRYPOINT ["ionic"]
11  CMD ["serve", "--all", "--port", "8100", "--livereload-port", "35729"]
```

*Ilustración 48: DockerFile de Ionic*
(Imagen propia, basada en la práctica)

- **Api Spring boot parte back-end**

Tal como lo recomienda la documentación oficial de Spring, se utilizó la imagen openjdk de DockerHub. Definimos una variable llamada "JAR_FILE", que tendrá el código fuente de la API y luego copiamos los archivos dentro de dicha variable al fichero "app.jar", dentro de la imagen. Para finalizar, exponemos el puerto que se utilizará y ejecutamos el archivo "app.jar" mediante el comando "java", definido en la instrucción "entrypoint", tal como se puede visualizar en la siguiente captura de pantalla:

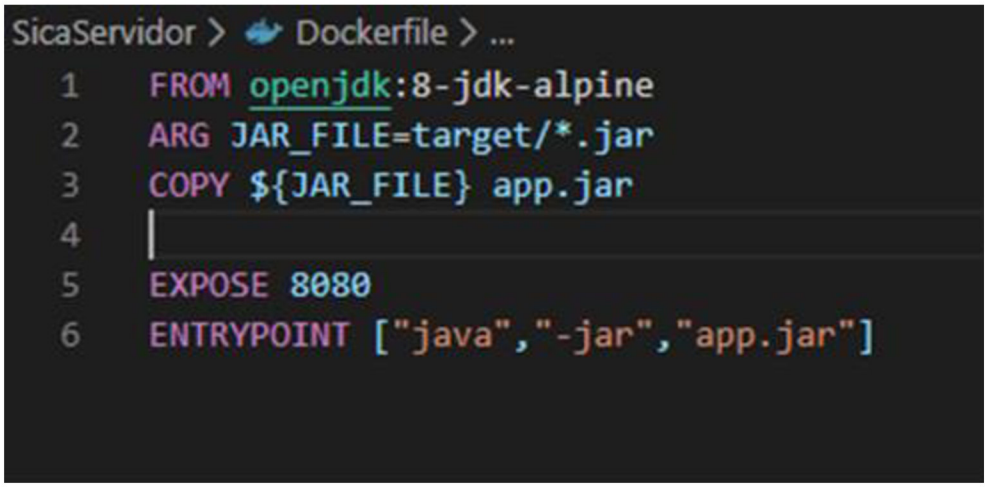



*Ilustración 49: DockerFile de la Api*
(Imagen propia, basada en la práctica)

Luego, se procede a construir las imágenes mediante el comando "Docker build –t <nombre imagen> <directorio DockerFile>". Y, una vez construidas, se suben a DockerHub mediante los comandos:

- "Docker login <usuario>"(es necesario generar un usuario de Docker Hub, para enlazarlo, en caso de no contar con uno).
- "Docker push <nombre de imagen>" (subimos al repositorio las imágenes creadas).

El resultado final de este proceso se muestra en la siguiente captura de pantalla:

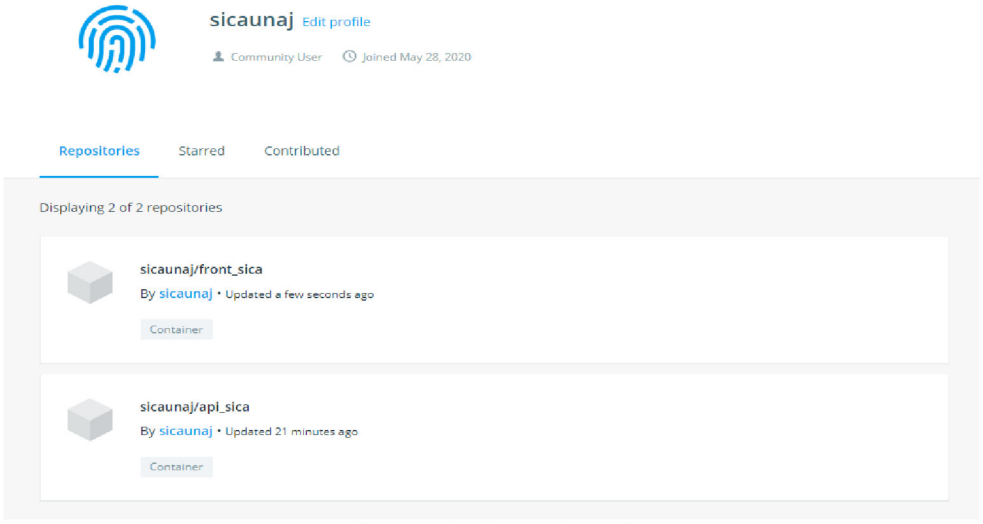



*Ilustración 50: DockerHub*
(Imagen propia, basada en la práctica)

En la ilustración 50, visualizamos las dos imágenes Docker creadas bajo el usuario "sicaunaj".

Por último, se codificó el archivo "docker-compose.yml", que posee la siguiente estructura:

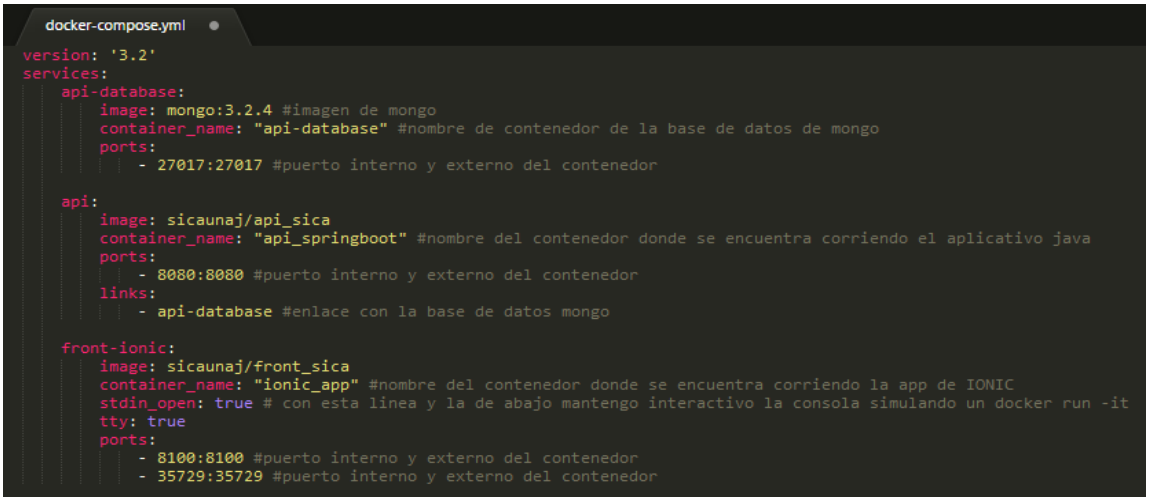



*Ilustración 51: Docker compose*
(Imagen propia, basada en la práctica)

Resulta interesante destacar que, para la base de datos Mongo, se utilizó la imagen del Docker oficial, en donde expusimos el puerto 27017 para utilizar. Luego, se definió la configuración para la API e Ionic, tal como se puede visualizar en la ilustración 51.

Finalmente, para inicializar todo el entorno, es necesario que, desde el lado del servidor de la universidad, se ejecute el comando "docker-compose up", ya que, al hacer esto, se realizará la construcción de las imágenes almacenadas en DockerHub y, además, se ejecutará el comando "run" para generar los contenedores necesarios para iniciar el servicio que dará funcionalidad al sistema desarrollado.

## 3. Conclusión

La presente Práctica Profesional Supervisada tuvo como objetivo principal desarrollar e implementar un sistema integral y escalable para el monitoreo de la calidad del agua en arroyos con el fin de elaborar planes de contingencia y reducir el riesgo de infecciones o enfermedades que se podrían llegar a generar debido a su contaminación.

Podemos decir que el sistema desarrollado cumple el requisito de ser escalable, ya que tiene la posibilidad de crecer sin necesidad de sufrir modificaciones, es decir, la cantidad de arroyos y de sondas que se pueden registrar no afecta el funcionamiento de la aplicación, ni tampoco afecta la cantidad de datos por municipios ingresados. Por lo tanto, cada uno de estos puede tener varios arroyos registrados y en análisis, los que, a su vez, tienen asignadas varias sondas multiparamétricas. De este modo, se logró cumplir con el objetivo de un diseño escalable.

Por otro lado, con respecto a la disponibilidad de los datos por analizar, concluimos que la información otorgada por las sondas es de fácil acceso para el usuario del sistema, si además consideramos que, adicionalmente, cuenta con un historial de las últimas cinco mediciones, a modo de registro, lo que le permite poder realizar un seguimiento del comportamiento de la contaminación de los arroyos.

Asimismo, en cuanto a la accesibilidad a los datos, se podría decir que se cumplió también con el requisito de diseñar un aplicativo multiplataforma, que puede ser utilizado en diferentes sistemas operativos y con distinta arquitectura (móvil o *desktop*).

Cabe destacar que el aplicativo fue desarrollado para que pueda extender sus funcionalidades a futuro, con el fin de mejorar su funcionamiento y rendimiento.

A nivel personal y profesional, adquirí experiencia en cuanto a la planificación y la gestión de un proyecto, la implementación de una metodología de desarrollo y de mejores prácticas para escribir un código. También, fue necesario y clave el entendimiento de un sistema externo para generar una mejora en cuanto a su funcionalidad, logrando adaptar así dos sistemas distintos con un objetivo en común.

La Práctica Profesional Supervisada presente nos brinda la posibilidad de realizar un proyecto de inicio a fin, en el que se ponen a prueba y se dinamizan, en sus potencialidades y límites, conceptos vistos, a lo largo de la carrera universitaria, y proyectados al “ámbito

laboral". Además, se tuvo que interactuar, por medio del tutor organizacional, con el Departamento de Sistemas de la Universidad Nacional Arturo Jauretche, para poder llevar adelante la implementación del sistema dentro de los servidores productivos, y también con docentes del departamento de Ciencias Sociales y Administración, encargados del proyecto a nivel global.

Resulta importante volver a insistir en que fue una experiencia muy satisfactoria durante la que logré comprender más cabalmente conceptos teóricos y prácticos, mediante un desarrollo "*full stack*" y a través del uso de varias tecnologías vigentes, con la finalidad de que funcionaran bajo un propósito en común, logrando así, en última instancia, satisfacer objetivos propios y de terceros.

Para concluir, con respecto al futuro de este sistema, existe la intención de escalar a nivel nacional e internacional, es decir, la expectativa y la pretensión de que se aplique en todos los arroyos de la Argentina (en primera instancia) y luego, en todo el mundo. La contaminación es un problema muy importante en todos los países y, desde mi punto de vista, todos los sistemas que sirvan para reducir este factor tienen altas posibilidades de crecer y ser exitosos, siempre con el fin de ayudar al bienestar y salud de los seres vivos. También, existe la intención de mejorar funciones y características del aplicativo, para lograr mayor precisión al registrar los datos y proporcionar métricas que permita que el usuario pueda realizar un análisis más eficiente y específico sobre la contaminación.

## Ilustraciones

## 4. Bibliografía

Codingpotions, documentación para desarrolladores (1 de mayo de 2018). Qué es Angular y sus ventajas. Cómo se instala. Recuperado de https://codingpotions.com/introduccion instalación-angular. [Consulta: 29 de octubre de 2019].

Docker Inc., guía para desarrolladores (s.f.). ¿Por qué Docker? Recuperado de https://www.docker.com/resources/what-container. [Consulta: 28 de octubre de 2019].

Esaú, A. (5 de mayo de 2014). Docker, Qué es y sus principales características. Recuperado de https://openwebinars.net/blog/docker-que-es-sus-principalescaracteristicas/. [Consulta: 28 de octubre de 2019].

Google Inc, documentación para desarrolladores (s.f.). Features & benefits. Recuperado de https://angular.io/features. [Consulta: 29 de octubre de 2019].

Manual web, documentación para desarrolladores (s.f.). ¿Qué es MongoDB? Recuperado de http://www.manualweb.net/mongodb/que-es-mongodb/. [Consulta: 31 de octubre de 2019].

MongoDB Inc., guía para desarrolladores (s.f.). What is MongoDB? Recuperado de https://www.mongodb.com/es/what-is-mongodb. [Consulta: 31 de octubre de 2019].

Muradas, M. Yanina (5 de junio de 2018). Conoce qué es Spring Framework y por qué usarlo. Recuperado de https://openwebinars.net/blog/conoce-que-es-spring-framework-y por-que-usarlo/. [Consulta: 29 de octubre de 2019].

Node.js, Inc., documentación para desarrolladores (sf). Acerca de Node.JS. Recuperado de https://nodejs.org/es/about/. [Consulta: 30 de noviembre de 2019]

Perry, J. Steven (11 de mayo de 2017). Aspectos básicos de Spring Boot. Recuperado de

https://www.ibm.com/developerworks/ssa/library/j-spring-boot-basicsperry/index.html.
[Consulta: 29 de octubre de 2019].