



QRSec: Sistema de control de acceso de invitados a barrios privados

Soria Gava, Lucas Damián. DNI: 42670460

Docente a cargo: Prof. Dra. Leiton, Ruth.

Tutor: Mag. Ing. Córdoba, Diego.

Cátedra: Trabajo Integrador Final III (TIF).

Carrera: Ingeniería en informática. 5° año.

Facultad de ingeniería - Sede Central

Universidad de Mendoza

22 de Junio de 2022

Índice:

4 Metodología:	3
4.1 Tecnologías usadas:	3
4.1.1 Estructura base del proyecto:	3
4.1.2 Diseño gráfico de la página:	3
4.1.3 Frontend:	3
4.1.4 Backend:	4
4.1.5 Base de datos:	6
4.1.6 Herramienta de control de versiones:	7
4.1.7 Virtualización:	9
4.1.8 Despliegue:	9
4.2 Desarrollo:	10
4.2.1 Diseño gráfico:	10
4.2.1.1 Sprint 1:	10
4.2.1.1.1 Definición de una paleta de colores:	10
4.2.1.1.2 Pantalla de creación de una invitación:	11
4.2.1.1.3 Pantalla para compartir la invitación:	12
4.2.1.1.4 Pantalla de creación de un Invitado nuevo:	12
4.2.1.1.5 Creación del logotipo de QRSec:	13
4.2.1.2 Sprint 2:	13
4.2.1.2.1 Pantalla de visualización de la invitación:	13
4.2.1.2.2 Pantalla de escaneo de invitaciones:	14
4.2.1.2.3 Pantalla de inicio de sesión:	14
4.2.1.2.4 Creación del icono de QRSec:	15
4.2.2 Desarrollo del Frontend:	15
4.2.2.1 Sprint 1:	15
4.2.2.1.1 Desarrollo de las pantallas de creación de invitación y compartir invitación:	15
4.2.2.1.2 Desarrollo de un la barra superior:	17
4.2.2.2 Sprint 2:	17
4.2.2.2.1 Desarrollo de la pantalla 'Agregar invitado':	17
4.2.2.2.2 Desarrollo de la pantalla de visualización de la Invitación:	18
4.2.2.2.3 Desarrollo de la pantalla de escaneo de Invitaciones:	19

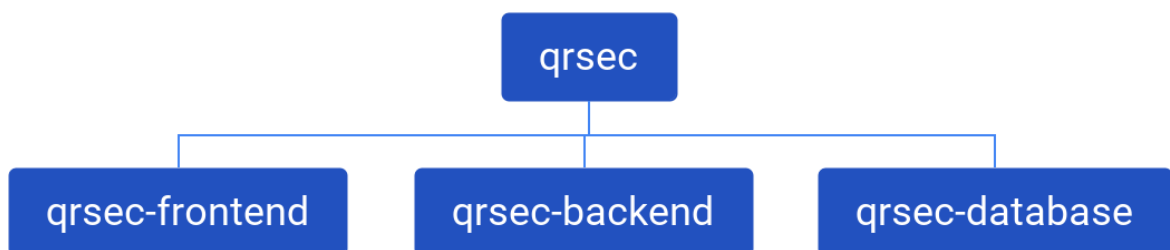
4.2.2.2.4 Desarrollo de la pantalla de inicio de sesión:	20
4.2.2.2.5 Modificación de la barra superior:	21
4.2.3 Desarrollo del Backend:	21
4.2.3.1 Sprint 1:	21
4.2.3.1.1 Asociación de la base de datos con una aplicación de Spring:	21
4.2.3.1.2 Creación de los modelos:	22
4.2.3.1.3 Creación de los primeros endpoint:	24
4.2.3.1.4 Configuración de la política de CORS:	25
4.2.3.2 Sprint 2:	26
4.2.3.2.1 Modificación de los modelos:	26
4.2.3.2.2 Inicio de sesión:	26
4.2.3.2.3 Modificación de endpoints:	26
4.2.4 Despliegue:	27
4.2.5 Creación de certificados SSL/TLS:	27
6 Referencias bibliográficas:	28

4 Metodología:

4.1 Tecnologías usadas:

4.1.1 Estructura base del proyecto:

El código se encuentra dividido en 3 directorios principales: frontend, backend y base de datos. De esta forma todo lo desarrollado de la aplicación web se encuentra contenida dentro de la capa a la que pertenece en el stack.



Imágen de elaboración propia

La relación entre estas tres capas se da también en ese orden, el frontend se relaciona con el backend a través de peticiones a la API desarrollada en este, y este a su vez se relaciona con la base de datos.

4.1.2 Diseño gráfico de la página:

Para tener una noción del diseño y la distribución del contenido de las páginas web que se desarrollaron, se usó la herramienta de diseño UX/UI [Marvel](#). El diseño se pensó desde un principio como una guía para el desarrollo y no como un estilo fijo, por lo tanto, las imágenes que se produjeron son aproximaciones del desarrollo final. Además, para que el desarrollo de las mismas siga un diseño de colores uniforme a lo largo de todas sus pantallas, se definió una paleta de colores, la cual se confeccionó con la ayuda de la herramienta web [coolors](#).

4.1.3 Frontend:

El frontend está principalmente desarrollado en el lenguaje de programación JavaScript. Junto con [npm](#) como su manejador de paquetes, para instalar las librerías necesarias. Las librerías más usadas fueron las de [React](#) para crear las

distintas funcionalidades de la aplicación web, Material UI ([mui](#)) para darle el estilo a las mismas, [react-qr-code](#) para poder crear el código QR de la invitación, [qrreader](#) para poder leer dicho QR, y por último, [react-google-maps/api](#) para poder mostrar un mapa que indique la ubicación de la casa a la que fue invitado.

El código principal se encuentra dentro del directorio “src”, dividido en los subdirectorios “components”, “data” y “pages”, cada uno contiene diferentes tipos de ficheros, dependiendo de las funciones que desempeñen.

El directorio “pages” contiene los componentes de React de más alto nivel, que simbolizan las páginas que los diferentes tipos de usuarios podrán ver. El directorio “components” contiene los elementos de código que *componen* a las páginas. El directorio “data” contiene todo el código que manipula información, principalmente llamadas a la API del backend.

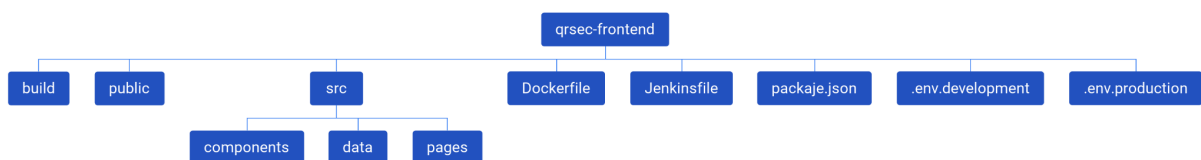
Las dependencias del proyecto se encuentran dentro del fichero “package.json”, el cual es generado automáticamente por *npm*.

Dentro de los archivos .env.development y .env.production, conocidos como archivos “dotenv”, se encuentran las constantes de la aplicación que son dependientes de los entornos de desarrollo y producción correspondientemente.

La carpeta “public” contiene principalmente los logos de la aplicación y los archivos de configuración de estilos visuales del frontend.

Por último, la versión de producción de la aplicación se encuentra en el directorio “build”.

Arbitrariamente para la programación del proyecto se decidió utilizar Visual Studio Code, uno de los IDEs más populares actualmente.



Imágen de elaboración propia

4.1.4 Backend:

El desarrollo del backend se basó en el lenguaje Java (versión 11), utilizando el framework de [Spring](#). Junto con [Maven](#), una herramienta de software para la

gestión y construcción de proyectos Java. La dependencia más importante fue [spring-boot-starter-data-mongodb](#), que posibilitó la comunicación del backend con la base de datos.

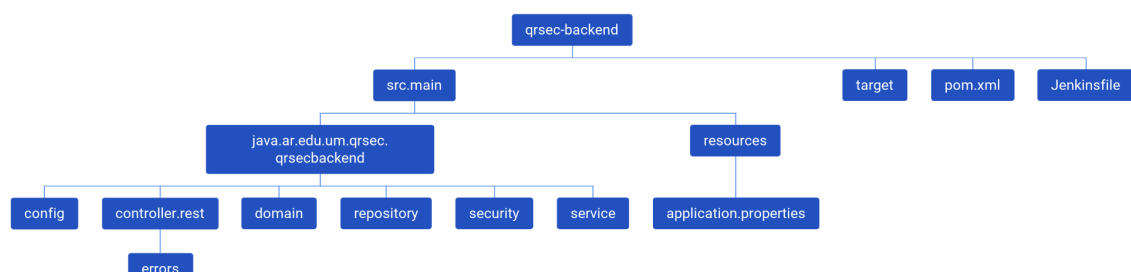
Dentro del directorio `qrsec-backend` nos encontramos con una estructura de archivos común para un proyecto de Spring. Los archivos principales en este directorio son: `pom.xml` y `application.properties`. El primer archivo contiene todas las dependencias del proyecto, y el segundo contiene variables de entorno, como en el frontend lo hacen los archivos `dotenv`.

La estructura de directorios elegida fue una compuesta por seis carpetas principales ubicadas dentro de `src`: `config`, `controller`, `domain`, `repository`, `security` y `service`. Dentro del directorio `config` se encuentran las clases encargadas de las configuraciones de la aplicación, como por ejemplo, las políticas de CORS o los permisos de acceso requeridos para cada endpoint. `domain` contiene las clases que representan los documentos de la base de datos y cómo se relacionan entre sí. `repository` contiene solicitudes más complejas a la base de datos o que no estén incluidas en la librería usada. `security` contiene clases relacionadas con la generación de JWTs. `service` contiene la lógica detrás de cada endpoint de la API. Por último, `controller` expone dichos endpoints, los métodos HTTP que permite y los permisos necesarios para acceder a estos (en términos de rol o autoridad del usuario).

El directorio `target` contiene entre otros archivos, la versión compilada para producción de la aplicación.

Para el testeo de la API desarrollada se utilizó [Insomnia](#), un cliente para APIs diseñado para el testing de las mismas.

Por último, el IDE seleccionado para el desarrollo fue IntelliJ IDEA, desarrollado por JetBrains y muy popular actualmente para el desarrollo en Java.



Imágen de elaboración propia

Las compilaciones de la aplicación tanto para el backend como para el frontend siguen el standard de versionado X.Y.Z, donde:

- X: Cambios mayores.
- Y: Cambios menores.
- Z: Solución de errores.

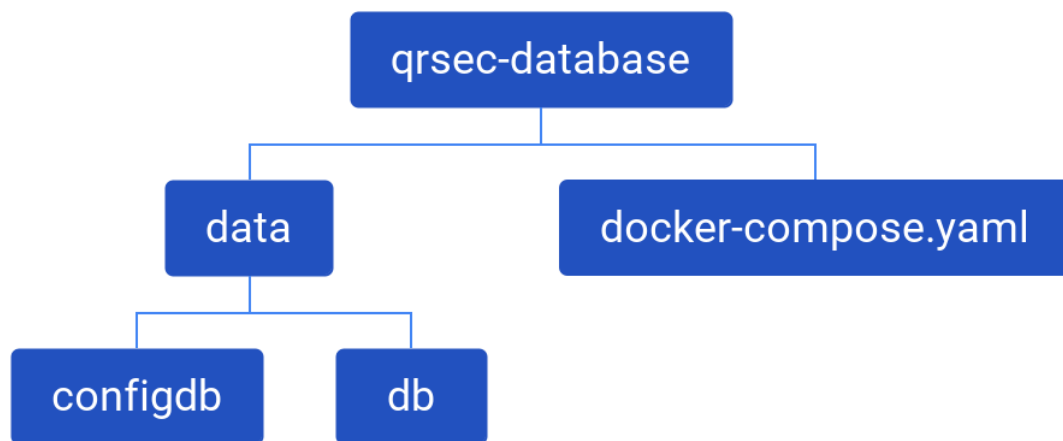
4.1.5 Base de datos:

El motor de base de datos utilizado para el proyecto es [MongoDB](#). Debido a que el sistema operativo en el que se desarrolló la aplicación está basado en una distribución de [Arch Linux](#), no es posible utilizar una base de datos local para el desarrollo. Por eso se decidió virtualizar la misma con Docker, a continuación se presenta el código para su configuración.

```
version: '3'
services:
  mongodb:
    image: mongo
    container_name: mongodb
    ports:
      - 27017:27017
    volumes:
      - /home/lucas/Documents/qrsec/qrsec-database/data/db:/data/db
      - /home/lucas/Documents/qrsec/qrsec-database/data/configdb:/data/configdb
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=password
    restart: unless-stopped
  mongo-express:
    image: mongo-express
    container_name: mongo-express
    ports:
      - 8081:8081
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=root
      - ME_CONFIG_MONGODB_ADMINPASSWORD=password
      - ME_CONFIG_MONGODB_SERVER=mongodb
    restart: unless-stopped
```

En este archivo `docker-compose.yaml` se puede apreciar la configuración de dos servicios, el primero es el motor de base de datos junto con la configuración de las credenciales necesarias para ingresar al mismo. El segundo servicio es un contenedor de *mongo-express*, el cual posibilita explorar las bases de datos presentes en el contenedor anterior a través de una interfaz gráfica en la web, especialmente útil para la etapa de desarrollo.

El primer servicio (mongodb) presenta dos volúmenes para poder persistir tanto las bases de datos como sus configuraciones, siguiendo el sistema de ficheros presentes en la siguiente imagen.



Imágen de elaboración propia

Como se puede observar, tanto la base de datos persistida como el archivo de configuración de los servicios están almacenados en el directorio `qrsec-database`, al que se hizo referencia en la sección 4.1.1.

Por último, cabe mencionar que si bien fue necesario utilizar Docker para virtualizar el motor de base de datos de MongoDB en la fase de desarrollo de la aplicación, en el servidor en producción no lo es. Esto es así ya que utiliza la versión 22.04 de Ubuntu, un sistema operativo basado en Debian, compatible con dicho motor de base de datos. A pesar de que no es necesario virtualizar la base de datos en Docker, por simplicidad se decidió mantener esta medida en el servidor de producción.

4.1.6 Herramienta de control de versiones:

Para el control de versiones se utilizó la herramienta [Git](#), junto con la plataforma de desarrollo colaborativo [GitHub](#).

El repositorio utilizado para almacenar el código del proyecto es:

<https://github.com/LucaSor1a/qrsec>



Fuente: <https://github.com/bryan2811/Curso-de-Git-y-Github-Platzi>

Para cada mensaje de *commit* se utilizó un estándar propio que es una simplificación de buenas prácticas comúnmente utilizadas en la industria para su generación. El mismo tiene la siguiente estructura:

<tipo>: <resumen>

Donde los **tipos** pueden ser:

- FEAT: Nuevas funcionalidades.
- FIX: Arreglos de fallas o bugs.
- REFACTOR: Cambios en la estructura de archivos del código o renombre de archivos.

El **resumen** es simplemente un texto que sintetiza y describe los cambios realizados en ese *commit*.

4.1.7 Virtualización:

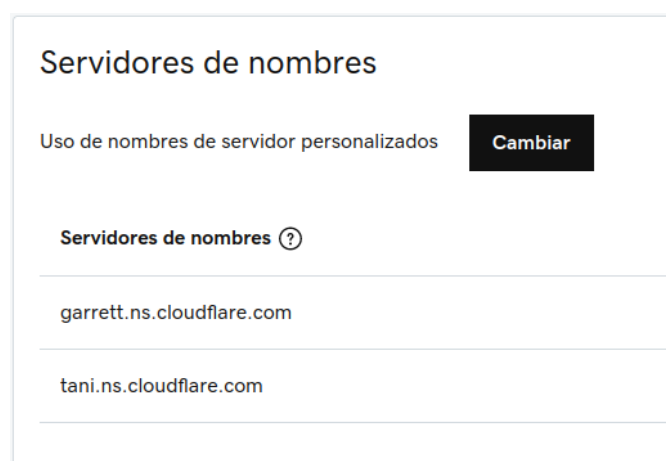
Como ya se mencionó en puntos anteriores, se decidió utilizar [Docker](#) para virtualizar la base de datos, cuya configuración ya fue detallada en la sección 4.1.5.

También se puede utilizar Docker para, en un futuro, virtualizar todo el sistema sin importar la plataforma sobre la que corra, creando imágenes tanto del backend como del frontend.





4.1.8 Despliegue:

Una vez que se crearon las versiones de producción del frontend y backend, se necesita un servidor desde el cual exponer la aplicación al público, para ello se decidió utilizar los servicios de Amazon, [AWS](#). Se eligió AWS porque ofrece un servicio de creación de instancias pequeñas gratuitas (mientras el consumo de recursos sea menor a un límite establecido) durante el primer año después de crear una cuenta. AWS ofrece la posibilidad de crear una instancia de un servidor Ubuntu 22.04 con 1Gb de memoria RAM y hasta 30Gb de almacenamiento, suficiente para las necesidades del proyecto.

Además de un servidor desde el cual distribuir la aplicación, se necesita un dominio y un DNS para que este sea fácilmente accesible. Como registrador de dominios se eligió contratar los servicios de [GoDaddy](#), el dominio comprado fue [Isoria.com](#). Por último, el DNS seleccionado fue [Cloudflare](#), ya que este ofrece sus servicios de forma gratuita. La configuración del DNS para el dominio mencionado anteriormente se muestra en las siguientes imágenes.



Configuración de DNS en GoDaddy (Imagen de elaboración propia)

Type	Name	Content	Proxy status	TTL	Actions
A	Isoria.com	54.227.162.119	 Proxied	Auto	Edit ▶
CNAME	_domainconnect	_domainconnect.gd.domainc...	 Proxied	Auto	Edit ▶
CNAME	edu	ec2-54-152-52-117.compute-1...	 Proxied	Auto	Edit ▶
CNAME	www	Isoria.com	 Proxied	Auto	Edit ▶

Configuración de DNS en Cloudflare (Imagen de elaboración propia)

Debido a que la IP pública que ofrece Amazon para conectarse al servidor creado puede variar, se utilizó la imagen de Docker [oznu/cloudflare-ddns](https://hub.docker.com/r/oznu/cloudflare-ddns/), que resulta muy práctica ya que modifica automáticamente la configuración del DNS de Cloudflare para que la IP de [Isoria.com](https://www.isoria.com) siempre sea la correcta.

```
version: '3'
services:
  cloudflare-ddns:
    image: oznu/cloudflare-ddns
    container_name: cloudflare-ddns
    restart: unless-stopped
    environment:
      - API_KEY=AT2qMl1WyYqrondfkbRTpGPVPRx1IaUrKOqPAMwc
      - ZONE=Isoria.com
      - PROXIED=true
```




4.2 Desarrollo:



4.2.1 Diseño gráfico:

4.2.1.1 Sprint 1:

4.2.1.1.1 Definición de una paleta de colores:

A partir de la herramienta de diseño mencionada en la sección 4.1.2, y siguiendo los significados de los colores de la *teoría de colores* para seguridad, se definieron cinco principales:

- #EBEBEB 
- #6FEC82 
- #FF101F 

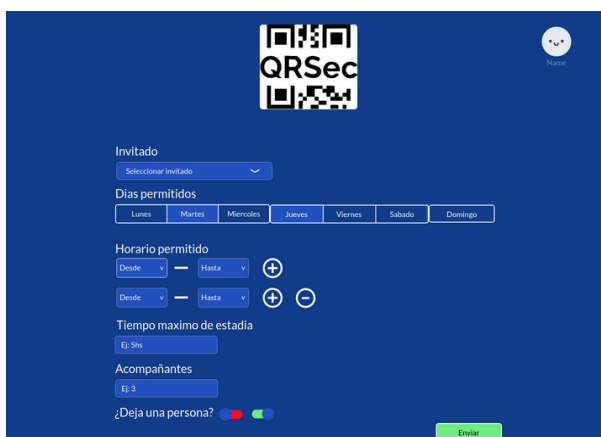
- #2251BF 
- #103C89 



Fuente: <https://coolors.co/ebebeb-6fec82-ff101f-2251bf-103c89>

4.2.1.1.2 Pantalla de creación de una invitación:

Estas imágenes corresponden a la secuencia de acciones para la generación de nuevas invitaciones. En ellas se incluye el comportamiento del botón desplegable para la selección de invitados, la selección múltiple de días de la semana y de rangos horarios y por último, los posibles colores que puede tomar el botón de tipo “switch”.



Pantalla base de creación de invitación



Botón desplegable de selección de invitados

Imágenes de elaboración propia

4.2.1.1.3 Pantalla para compartir la invitación:

Es posible visualizar esta pantalla una vez que se genera una nueva invitación. Contiene el link que luego el invitado debe visitar para poder visualizar su invitación y un botón para copiar el mismo al portapapeles.



Imagen de elaboración propia

4.2.1.1.4 Pantalla de creación de un Invitado nuevo:

Esta pantalla se puede ver cuando se hace click en 'Agregar Invitado' dentro del menú de selección de invitados. Permite agregar un nuevo invitado que esté relacionado al usuario que lo crea.

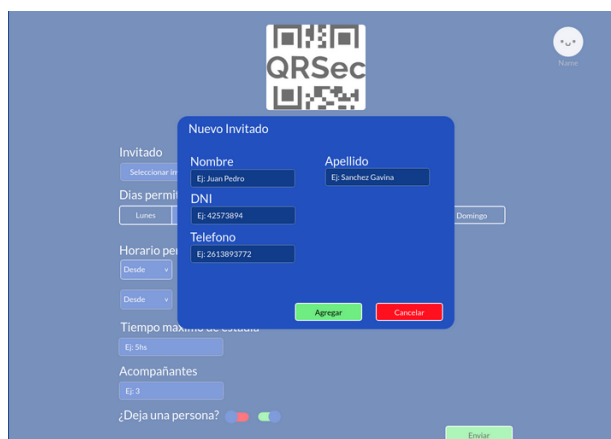


Imagen de elaboración propia

4.2.1.1.5 Creación del logotipo de QRSec:

Por último, se generó el logotipo que identifica al desarrollo y evidencia su característica principal. La imagen que se presenta a continuación está formada por un código QR que contiene en su interior la palabra QRSec y que a su vez este está atravesado por dicha palabra. La imagen fue vectorizada y presenta un fondo transparente.



Imagen de elaboración propia

4.2.1.2 Sprint 2:

4.2.1.2.1 Pantalla de visualización de la invitación:

Esta es la pantalla a la que tiene acceso el invitado, en ella se puede ver el código QR que representa su invitación y un mapa que señala como llegar hasta la casa a la que fue invitado.

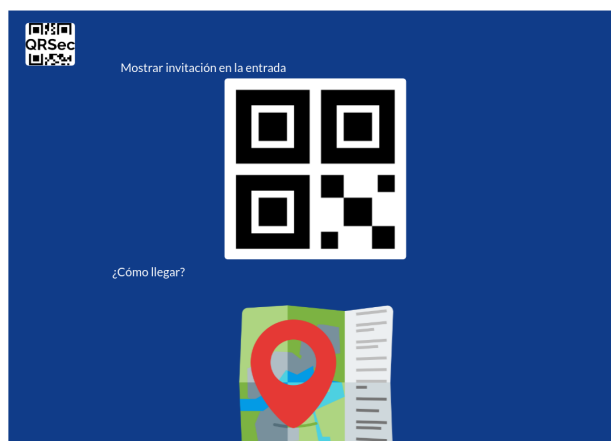
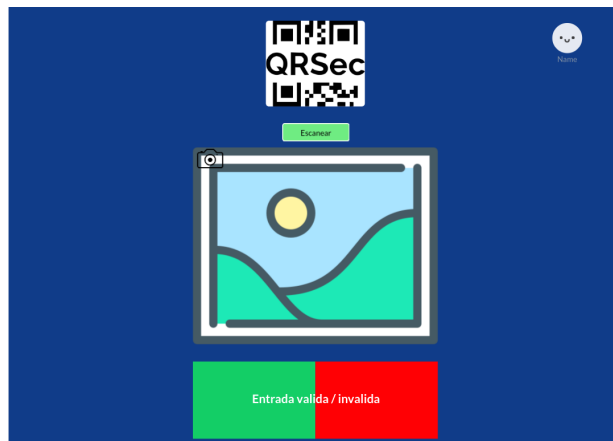


Imagen de elaboración propia

4.2.1.2.2 Pantalla de escaneo de invitaciones:

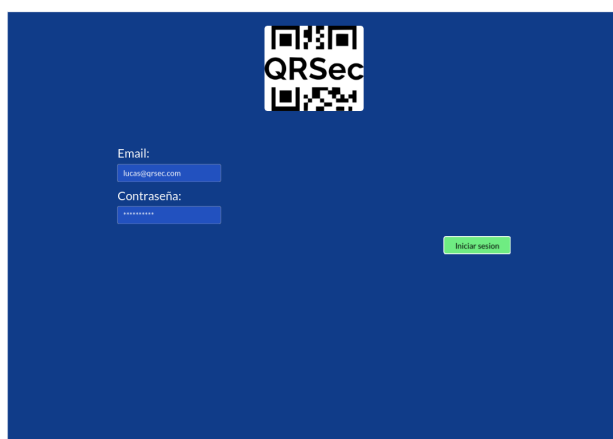
Esta es la pantalla a la que tendrá acceso el guardia de la entrada del barrio. Está compuesta de una sección que muestra la cámara de la barrera y debajo un recuadro que informa si la invitación que detecta es válida o inválida y los datos que esta contiene la misma en caso de ser válida.



Imágen de elaboración propia

4.2.1.2.3 Pantalla de inicio de sesión:

En esta última pantalla se encuentra un simple formulario para ingresar los datos de inicio de sesión.



Imágen de elaboración propia

4.2.1.2.4 Creación del icono de QRSec:

La última pieza gráfica que se generó fue un pequeño ícono llamado *favicon* el cual se muestra en la esquina superior izquierda de la pestaña del navegador, junto con el título de la página.



Imagen de elaboración propia

Debido a que su tamaño es mucho menor al resto de imágenes presentadas en la página, no fue posible utilizar el logotipo de la sección 4.2.1.1.5 y para que pueda ser fácilmente distinguible, se optó por la simpleza que representa utilizar las letras *QR* sobre un fondo negro.

4.2.2 Desarrollo del Frontend:

4.2.2.1 Sprint 1:

4.2.2.1.1 Desarrollo de las pantallas de creación de invitación y compartir invitación:

Luego de realizar el diseño de las primeras pantallas se decidió utilizar para el desarrollo del frontend la librería de Material UI (como ya se especificó en la sección 4.1.3) ya que ésta ofrece componentes de React muy útiles y estilizados que son fáciles de modificar para que se ajusten a las necesidades del proyecto. Para poder utilizar dicha librería se consultó la documentación de la misma y se complementó con tutoriales básicos de YouTube respecto a dichas tecnologías para dar un comienzo más ameno al proceso de aprendizaje. Dichos videos fueron: [un tutorial de React](#) y [un tutorial de mui](#).

Las modificaciones al estilo de los componentes provistos por la librería de Material UI se encuentran dentro del directorio *public*, como se detalla en la sección 4.1.3.

Finalmente se logró implementar una página capaz de:

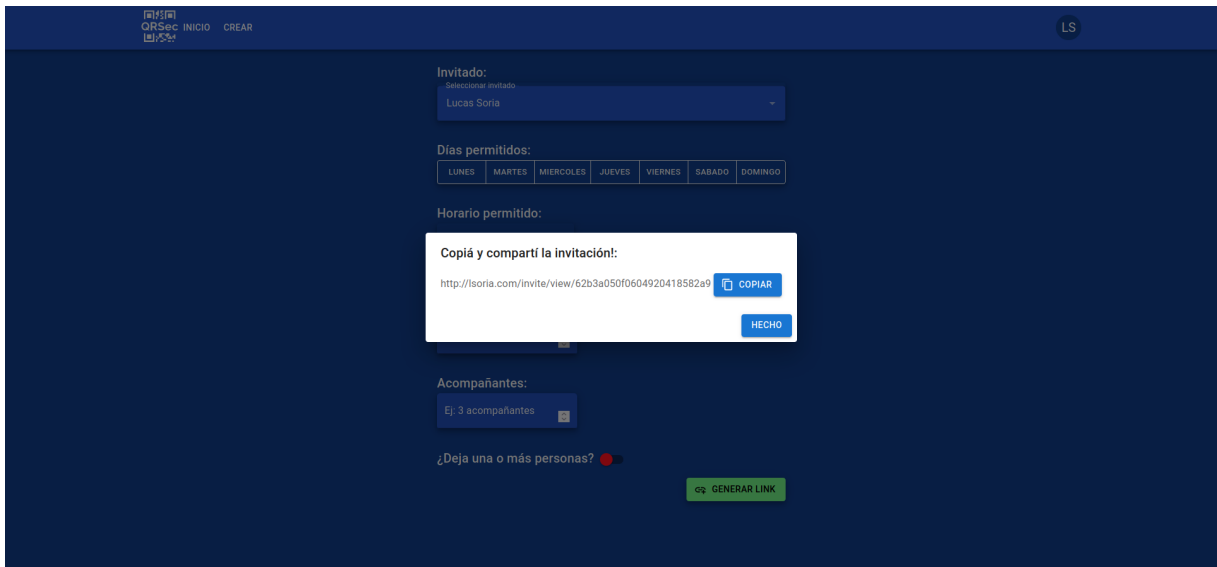
- Tomar la lista de Invitados almacenada en la base de datos a través de la API y mostrar sus nombres completos para su posterior selección.
- Hacer una selección múltiple de los días de la semana que esa persona tiene permitido ingresar al barrio [opcional].
- Hacer una selección de los rangos horarios donde tiene permitido ingresar al barrio [opcional].
- Establecer un tiempo máximo de estadía una vez que ingresó al barrio [opcional].
- Fijar la cantidad de personas que ingresan con el Invitado [opcional].
- Establecer si dicho Invitado en realidad ingresa al barrio para dejar a una persona, por ejemplo, si es un padre que viene a dejar a su hijo en la casa de un amigo.

The screenshot shows the 'Crear' (Create) page of the QRSec application. The interface is dark blue with white text. At the top, there's a navigation bar with 'QRSec', 'INICIO', and 'CREAR' links, and a user profile icon labeled 'MT'. The main form contains the following fields:

- Invitado:** A dropdown menu showing 'Seleccionar invitado' and 'Lucas Damián Soria Gava'.
- Días permitidos:** A row of buttons for 'LUNES', 'MARTES', 'MIÉRCOLES', 'JUEVES', 'VIERNES', 'SABADO', and 'DOMINGO'.
- Horario permitido:** A grid of time slots: '15:30', '18:30', '20:00', and '22:45', each with a plus/minus icon. There are also '+' and '-' icons to the right of the grid.
- Tiempo máximo de estadía:** A text input field with 'Ej: 5hs' and a numeric input field with '3'.
- Acompañantes:** A text input field with 'Ej: 3 acompañantes' and a numeric input field with '3'.
- ¿Deja una o más personas?** A checkbox with a red dot.
- GENERAR LINK:** A green button at the bottom right.

Imágen de elaboración propia

Una vez que se completan los campos deseados, se oprime el botón de 'Generar link' y a través de la API se genera una nueva Invitación y se muestra un diálogo que permite copiar el link con la Invitación para compartir con el Invitado, como se muestra a continuación.



Imágen de elaboración propia

4.2.2.1.2 Desarrollo de un la barra superior:

Uno de los cambios más grandes que se pueden apreciar con respecto al diseño original es la introducción de una barra superior que permite navegar a las distintas páginas de la aplicación. Además de poseer dichos enlaces, la misma muestra en todo momento el ícono de QRSec y un ícono (por el momento sin funcionalidad) que muestra las primeras dos iniciales del usuario que esté utilizando la página en ese momento.

4.2.2.2 Sprint 2:

4.2.2.2.1 Desarrollo de la pantalla 'Agregar invitado':

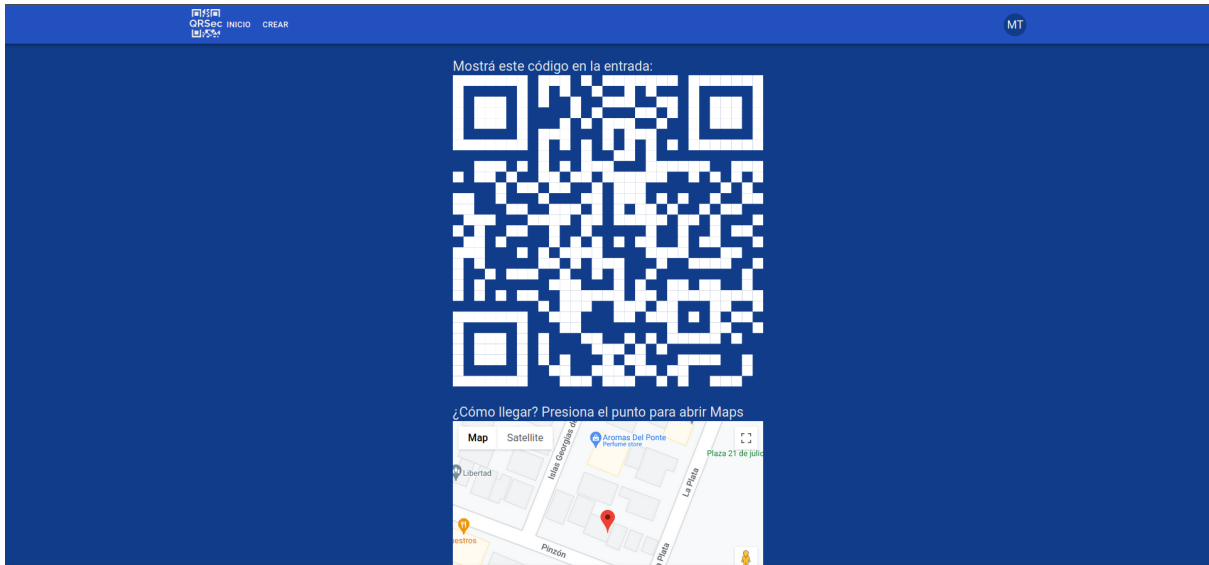
Durante el segundo Sprint se desarrolló la funcionalidad de 'Agregar invitado' dentro del menú de selección de Invitados. Si esta opción es seleccionada se despliega un diálogo que permite completar la información pertinente para generar un nuevo invitado. Una vez se completan los campos y se presiona el botón de 'crear', la página se comunica con el backend a través de la API y este registra dicho invitado en la base de datos. Así la próxima vez que se abra el menú desplegable, este estará disponible.

Imágen de elaboración propia

4.2.2.2.2 Desarrollo de la pantalla de visualización de la Invitación:

Para el desarrollo de la pantalla de visualización se hizo uso de la librería de react-qr-code que se describió anteriormente. Esta permite que a través de un texto, en este caso el id de la invitación, se genere un código QR que puede ser configurado según la necesidad del usuario. En este caso se necesitó configurar el código para que tenga el máximo nivel de tolerancia a errores, tal como lo describe Peter Kieseberg, et al. (2010) en su paper 'QR Code Security'. El máximo nivel es el H y permite una tolerancia al error de hasta el 30%, esto resulta especialmente útil para cuando se requiere una lectura más confiable y no es importante perder capacidad de codificación. En el caso de QRSec, el largo de la información a codificar es siempre de 24 caracteres, por lo que no se corre riesgo de perder capacidad de codificación.

Adicionalmente debajo del código QR se localiza un mapa con la dirección de la casa marcada, en caso de que el invitado nunca haya asistido a la misma. La integración del mismo a la página ha sido un problema inesperado ya que para hacerlo se debe crear una cuenta en el servicio de Cloud de Google (GCP) y se debe generar un token de acceso a la API de Maps de Google. La misma a su vez debe ser consumida a través de la librería de react-google-maps/api. Cabe notar que después de una cantidad preestablecida de peticiones a la API de Google por mes, este empieza a cobrar por su uso.



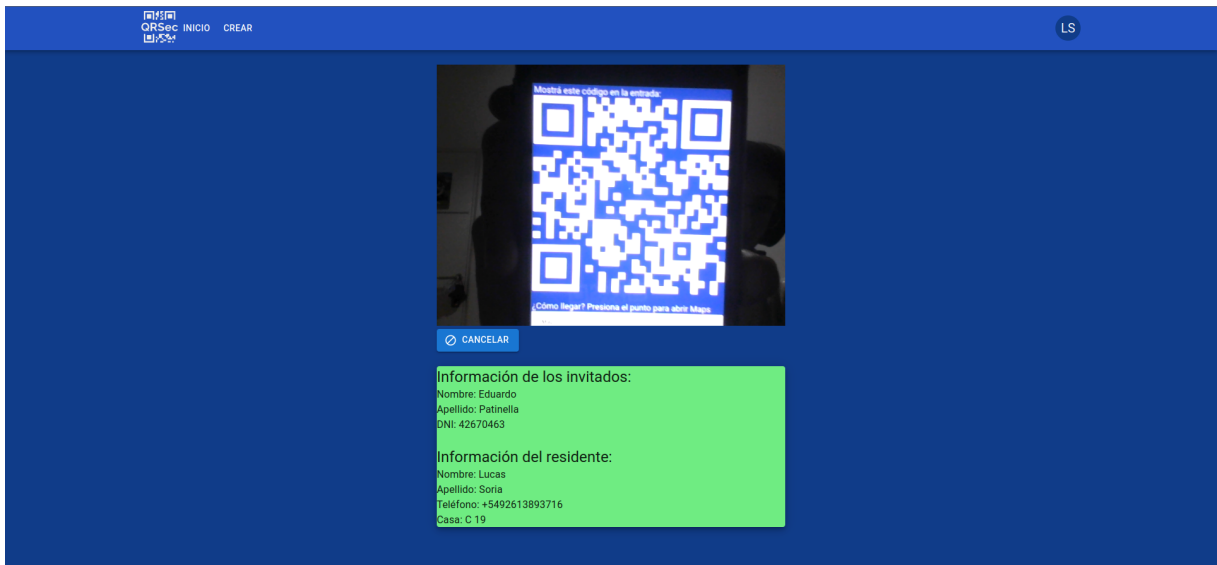
Imágen de elaboración propia

4.2.2.2.3 Desarrollo de la pantalla de escaneo de Invitaciones:

Después de haber podido generar la página de visualización de los códigos QR, se desarrolló la página de escaneo de los mismos, es a través de ella que se pueden validar las entradas.

Luego de realizar un escaneo exitoso del código, se hace una solicitud a la API con el id de la Invitación, en caso de que no exista esa entrada en la base de datos se envía un error 404 y la entrada es inválida. En el caso que la Invitación exista, esta se envía al frontend y se analiza la validez de sus campos, buscando que todos se cumplan, como por ejemplo, que la persona esté intentando ingresar dentro del horario permitido o en alguno de los días que tiene permitido ingresar.

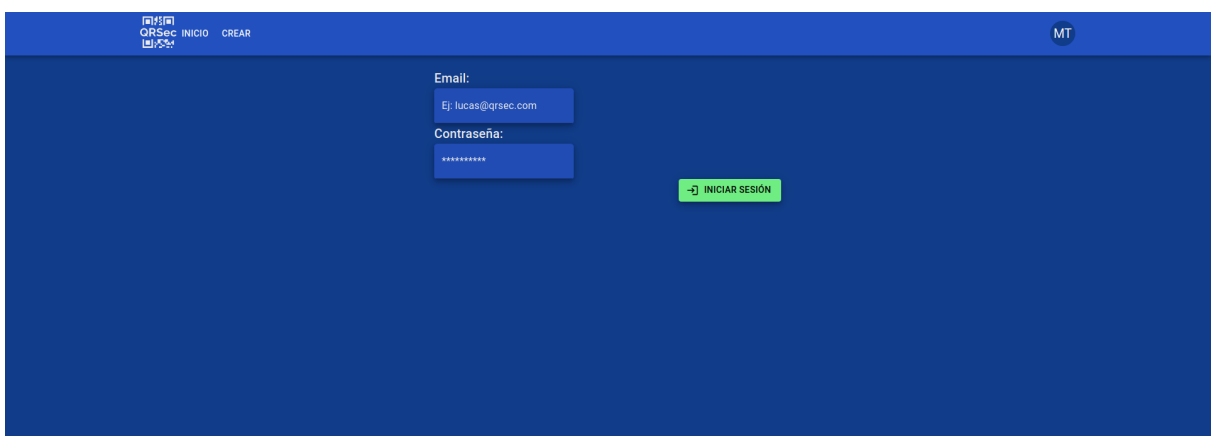
El escaneo del código resultó especialmente desafiante ya que las librerías que se probaron para realizar este proceso contenían todas problemas de compatibilidad ya sea con la versión de React que se utilizó (React 18), con el navegador o el sistema operativo. Por esta razón, se optó por utilizar una librería que implemente en el nivel más bajo posible dicha funcionalidad, utilizando simplemente Javascript y HTML5, para que a pesar de que se pierde potencia, se mantiene un sistema confiable.



Imágen de elaboración propia

4.2.2.2.4 Desarrollo de la pantalla de inicio de sesión:

Esta es una pantalla simple que contiene un pequeño formulario como se planteó en la sección 4.2.1.2.3 cuya funcionalidad es la de obtener el JWT (JSON Web Token) del usuario que utilizará la aplicación y guardarlo dentro del almacenamiento local del dispositivo que haga uso de la página, para que después el mismo tenga acceso a datos relevantes de la API para el usuario. Adicionalmente, si el usuario no ha iniciado sesión e intenta usar la página de escaneo de códigos QR o la página de generación de invitaciones, será redirigido a la página de inicio de sesión.



Imágen de elaboración propia

4.2.2.2.5 Modificación de la barra superior:

La última modificación realizada al frontend fue la de agregar funcionalidad real al icono de la barra superior. Se lo modificó de forma tal que enseña las iniciales del usuario que inició sesión y además, el menú que despliega permite cerrar la sesión, eliminando los JWT guardados en el almacenamiento local del dispositivo.

4.2.3 Desarrollo del Backend:

4.2.3.1 Sprint 1:

4.2.3.1.1 Asociación de la base de datos con una aplicación de Spring:

Para poder comunicar la base de datos con la aplicación de Spring se necesitó principalmente realizar la configuración de la misma dentro de las variables de entorno que se encuentran en el archivo `application.properties`, descrito en la sección 4.1.4. Dicha configuración es:

```
spring.data.mongodb.authentication-database=admin
spring.data.mongodb.username=root
spring.data.mongodb.password=password
spring.data.mongodb.database=qrsec
spring.data.mongodb.port=27017
spring.data.mongodb.host=localhost
spring.data.mongodb.auto-index-creation=true

spring.security.user.name=root
spring.security.user.password=password

api.path=/api
api.path.invite=/invites
api.path.guest=/guests
api.path.user=/users
api.path.login=/login
api.path.refresh=/token/refresh

server.port=8080
```

Dichas variables concuerdan con la configuración de la base de datos definida en la sección 4.1.5. La configuración presentada sigue la guía de la [documentación](#) de MongoDB respecto a Spring y la de 'AmigosCode' en su [tutorial](#) de YouTube.

4.2.3.1.2 Creación de los modelos:

Para poder relacionar un “documento” de la base de datos NoSQL y la aplicación de Spring, se debe crear una clase *modelo* que represente dicho documento. Inicialmente se plantearon siete clases, a continuación se listan mostrando su estereotipo (Document, Class), nombre, atributos y tipo de dato:

<Document> User	
email	String (Indexed)
password	String
authority	Set<Role>
firstName	String
lastName	String
dni	String (Indexed)
address	Address (DocumentReference)
phone	String

<Class> Role	
authority	String (ADMIN, OWNER, GUARD)

<Document> Guest	
firstName	String
lastName	String
dni	String (Indexed)
phone	String
owner	Set<User>

<Document> Invite	
owner	User (DocumentReference)
guests	Set<Guest> (DocumentReferece)
days	Set<String>
hours	List<List<String>>
maxTime	Integer
passengers	Integer
drop	Boolean
arrival	LocalDateTime
departure	LocalDateTime
created	LocalDateTime
modified	LocalDateTime

<Document> Address	
street	String
number	Integer
house	House (Indexed)
location	Location

<Class> House	
block	String
house	Integer

<Class> Location	
type	String
coordinates	List<Double>

Es importante destacar que cada clase con el estereotipo de documento posee un atributo adicional: *id*.

La última clase especificada (Location) sigue las especificaciones para una petición geoespacial al motor de base de datos MongoDB, según lo especifica su [documentación](#).

4.2.3.1.3 Creación de los primeros endpoint:

Una vez se crearon las clases que se relacionan con la base de datos, fue importante definir una clase *Repository* por cada una de ellas, en el caso de QRSec se crearon dichos *Repository* para las clases: *Address*, *Guest*, *Invite* y *User*.

Posteriormente, se definieron las clases *Controller* para *Guest* e *Invite*. Específicamente se crearon los endpoint de */api/guest* y */api/invite*, donde al primero se le definió el método GET, para que la pantalla creación de invitación pueda listar los Invitados presentes en la base de datos. Al segundo endpoint se le definió el método POST para que la pantalla de creación de invitaciones pueda efectivamente generar una nueva Invitación. Para cada método HTTP se creó también la lógica de negocio dentro de su correspondiente método en las clases *Service* pertinentes.

Durante el desarrollo de estos endpoint surgieron problemas de mapeo entre las clases y sus respectivos DTOs (Data Transfer Objects), por lo que se decidió omitir temporalmente el desarrollo de los mismos, al no ser una parte indispensable del desarrollo de la aplicación.

4.2.3.1.4 Configuración de la política de CORS:

Un problema encontrado durante el desarrollo del frontend fue la política de CORS, la cual impedía poder hacer peticiones a la API desde la página web. Este pudo superarse gracias a la [documentación](#) de Spring y el tutorial de [Baeldung](#) al respecto, culminando en la creación de la Clase detallada a continuación que se encuentra dentro del directorio *config*, mencionado anteriormente.

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Value("${api.path}")
    private String path;

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping(this.path +
            "/*").allowedOrigins("http://localhost", "https://localhost",
            "http://localhost:8080", "http://localhost:3000");
    }
}
```

4.2.3.2 Sprint 2:

4.2.3.2.1 Modificación de los modelos:

Debido a que los índices generados por Spring se propagan entre documentos, existían fallos a la hora de generar invitados asociados a distintos usuarios. Esto se pudo corregir cambiando dichos índices para que se generen sólo sobre aquellos documentos que posean el atributo que se está buscando indexar.

Además de esto, se modificaron los documentos de los Invitados, para que los mismos estén relacionados a un conjunto de Residentes. Este cambio se hizo teniendo en cuenta que cuando se implemente el inicio de sesión, interesa traer de la base de datos solamente aquellos invitados que estén relacionados al usuario que está usando actualmente el sistema.

4.2.3.2.2 Inicio de sesión:

Se programó el sistema de generación de JWT para poder relacionar automáticamente la generación de las Invitaciones o de nuevos Invitados al usuario que está realizando dichas peticiones a la API, así como para obtener de la misma información relevante para el usuario actual.

Este proceso fue muy laborioso e introdujo varios imprevistos. Para poder definir esta parte del código, fue necesario que primero se crearan filtros de autenticación y de autorización, además de cambios en las políticas de CORS anteriormente delimitadas, entre otras cosas.

Como resultado final, cuando un usuario ingresa sus credenciales en el endpoint `/login` el sistema le devuelve un *token de acceso* y otro de *refresco*, cuya utilidad es la de volver a generar un *token de acceso* sin la necesidad de volver a ingresar las credenciales.

Para este desarrollo resultaron especialmente útiles las guías de [‘Amigos Code’](#) y [Baeldung](#) sobre el tema.

4.2.3.2.3 Modificación de endpoints:

Ya que durante el primer Sprint solo se generaron los endpoints básicos para poder leer los invitados de la base de datos y generar nuevas invitaciones, en este Sprint se modificaron los mismos para admitir nuevos métodos y hacer que la generación tanto de Invitados como de Invitaciones asocie

automáticamente estos al usuario que las creó. Además, se agregaron dos nuevos controladores, uno para los usuarios (POST y GET) y otro para la generación del JWT. Por último, debido a la configuración generada en el punto anterior, ahora los endpoint o métodos pueden ser restringidos por rol o autoridad del usuario.

4.2.4 Despliegue:



En esta etapa se utilizaron las configuraciones detalladas en la sección 4.1 y adicionalmente se configuró [Nginx](#) para el despliegue del frontend tanto en el puerto 80 como en el puerto 443. Se necesita un despliegue en un entorno seguro (https) porque es un requisito para el uso de cámaras a través de una página web, como para el uso del portapapeles en el caso del botón de copia de la dirección de la invitación.

Tanto el backend como el motor de base de datos requieren menor esfuerzo, ya que estos simplemente deben ser ejecutados, el backend se ejecuta a través del archivo .jar generado a partir del código fuente y el motor de base de datos se ejecuta a través de su archivo docker-compose.yaml.

4.2.5 Creación de certificados SSL/TLS:

Finalmente y como se adelantó en la sección anterior, para poder generar un entorno seguro que permita utilizar la cámara para escanear las entradas, se utilizaron los certificados SSL/TLS generados a partir de la herramienta [Let's Encrypt](#). Tanto los para la certificación con *Let's Encrypt*, como el despliegue con *Nginx*, resultaron particularmente valiosas las guías *Digital Ocean*.

6 Referencias bibliográficas:

1. Adding Custom Environment Variables | Create React App. (s/f). Recuperado el 21 de junio de 2022, de <https://create-react-app.dev/docs/adding-custom-environment-variables/>
2. Advanced Load Balancer, Web Server, & Reverse Proxy. (s/f). NGINX. Recuperado el 22 de junio de 2022, de <https://www.nginx.com/>
3. Amigoscode. (2021a, junio 1). Spring Boot Tutorial—Build a Rest Api with MongoDB. <https://www.youtube.com/watch?v=ssj0CGxv60k>
4. Amigoscode. (2021b, julio 25). Spring Boot and Spring Security with JWT including Access and Refresh Tokens . <https://www.youtube.com/watch?v=VVn9OG9nfH0>
5. Arch Linux. (s/f). Recuperado el 20 de junio de 2022, de <https://archlinux.org/>
6. baeldung. (2017, enero 30). CORS with Spring | Baeldung. <https://www.baeldung.com/spring-cors>
7. baeldung. (2018, enero 20). Introduction to Spring Method Security | Baeldung. <https://www.baeldung.com/spring-security-method-security>
8. Build software better, together. (s/f). GitHub. Recuperado el 20 de junio de 2022, de <https://github.com>
9. Carlos Azaustre - Aprende JavaScript. (2021, abril 23). APRENDE REACT BÁSICO en 30 MINUTOS —Tutorial de React.js Desde Cero. <https://www.youtube.com/watch?v=EMk6nom1aS4>
10. Cloud Computing Services—Amazon Web Services (AWS). (s/f). Amazon Web Services, Inc. Recuperado el 20 de junio de 2022, de <https://aws.amazon.com/>
11. Cloudflare—The Web Performance & Security Company. (s/f). Cloudflare. Recuperado el 20 de junio de 2022, de <https://www.cloudflare.com/en-gb/>
12. ¡Compra dominio y hosting, y haz tu página web con el proveedor #1! GoDaddy AR. (s/f). GoDaddy. Recuperado el 20 de junio de 2022, de <https://www.godaddy.com/es>
13. Create a Palette—Coolers. (s/f). Coolers.Co. Recuperado el 20 de junio de 2022, de <https://coolers.co/ebebeb-6fec82-ff101f-2251bf-103c89>

14. Enabling Cross Origin Requests for a RESTful Web Service. (s/f). Recuperado el 20 de junio de 2022, de <https://spring.io/guides/gs/rest-service-cors/>
15. Geospatial Queries—MongoDB Manual. (s/f). Recuperado el 20 de junio de 2022, de <https://www.mongodb.com/docs/manual/geospatial-queries/>
16. Git. (s/f). Recuperado el 20 de junio de 2022, de <https://git-scm.com/>
17. Home—Docker. (2022, mayo 10). <https://www.docker.com/>
18. How To Deploy a React Application with Nginx on Ubuntu 20.04 | DigitalOcean. (s/f). Recuperado el 22 de junio de 2022, de <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-react-application-with-nginx-on-ubuntu-20-04>
19. How To Install Nginx on Ubuntu 20.04 | DigitalOcean. (s/f). Recuperado el 22 de junio de 2022, de <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04>
20. How To Secure Nginx with Let's Encrypt on Ubuntu 20.04 | DigitalOcean. (s/f). Recuperado el 22 de junio de 2022, de <https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04>
21. Let's Encrypt. (s/f). Recuperado el 22 de junio de 2022, de <https://letsencrypt.org/>
22. Marvel—The design platform for digital products. Get started for free. (s/f). Recuperado el 20 de junio de 2022, de <https://marvelapp.com/>
23. Maven – Welcome to Apache Maven. (s/f). Recuperado el 20 de junio de 2022, de <https://maven.apache.org/>
24. Maven Repository: Org.springframework.boot » spring-boot-starter-data-mongodb. (s/f). Recuperado el 20 de junio de 2022, de [https://mvnrepository.com/artifact/org.springframework.boot/spring-bo
ot-starter-data-mongodb](https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-mongodb)
25. MongoDB | Build Faster. Build Smarter. (s/f). MongoDB. Recuperado el 20 de junio de 2022, de <https://www.mongodb.com>
26. MUI: The React component library you always wanted. (s/f). Recuperado el 20 de junio de 2022, de <https://mui.com/>

- 27.Npm. (s/f). Recuperado el 20 de junio de 2022, de <https://www.npmjs.com/>
- 28.oznu/cloudflare-ddns—Docker Image | Docker Hub. (s/f). Recuperado el 22 de junio de 2022, de <https://hub.docker.com/r/oznu/cloudflare-ddns>
- 29.Qrreader. (s/f). Npm. Recuperado el 22 de junio de 2022, de <https://www.npmjs.com/package/qrreader>
- 30.React – A JavaScript library for building user interfaces. (s/f). Recuperado el 20 de junio de 2022, de <https://reactjs.org/>
- 31.@react-google-maps/api. (s/f). Npm. Recuperado el 22 de junio de 2022, de <https://www.npmjs.com/package/@react-google-maps/api>
- 32.React-qr-code. (s/f). Npm. Recuperado el 20 de junio de 2022, de <https://www.npmjs.com/package/react-qr-code>
- 33.Spring Boot Integration With MongoDB Tutorial. (s/f). MongoDB. Recuperado el 20 de junio de 2022, de <https://www.mongodb.com/compatibility/spring-boot>
- 34.Spring Boot—Application Properties. (s/f). Recuperado el 21 de junio de 2022, de https://www.tutorialspoint.com/spring_boot/spring_boot_application_properties.htm
- 35.Spring makes Java simple. (s/f). Spring. Recuperado el 20 de junio de 2022, de <https://spring.io/>
- 36.The API Design Platform and API Client. (s/f). Recuperado el 22 de junio de 2022, de <https://insomnia.rest/>
- 37.Traversy Media. (2020, octubre 9). Material UI React Tutorial. <https://www.youtube.com/watch?v=vyJU9efvUtQ>