Neiva, 31 de mayo de 2023

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Johan Sebastián Gutiérrez Repizo , con C.C. No. 1.003.819.455,

Juan Pablo Penagos Losada , con C.C. No. 1.008.392.415,

Autor(es) de la tesis y/o trabajo de grado titulado **APLICACIÓN WEB PARA IDENTIFICACIÓN DE TAXIS CON TECNOLOGÍA QR**

presentado y aprobado en el año 2023 como requisito para optar al título de

INGENIERO ELECTRÓNICO;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE: Johan Sebastián Gutiérrez Repizo

Firma: 

EL AUTOR/ESTUDIANTE: Juan Pablo Penagos Losada

Firma: 

TÍTULO COMPLETO DEL TRABAJO: APLICACIÓN WEB PARA IDENTIFICACIÓN DE TAXIS CON TECNOLOGÍA QR

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Gutiérrez Repizo	Johan Sebastián
Penagos Losada	Juan Pablo

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Quintero Polanco	Jesús David

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
No Aplica	No Aplica

PARA OPTAR AL TÍTULO DE: INGENIERO ELECTRÓNICO

FACULTAD: **INGENIERÍA**

PROGRAMA O POSGRADO: INGENIERÍA ELECTRÓNICA

CIUDAD: NEIVA **AÑO DE PRESENTACIÓN:** 2023 **NÚMERO DE PÁGINAS: 45**

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas _X_ Fotografías_X_ Grabaciones en discos___ Ilustraciones en general_X_ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. Aplicación Web	Web application	6. Taxis	Taxis
2. QR	QR	7. Node.js	Node.js
3. Mongo DB	Mongo DB	8. Transporte	Transport
4. Stack MERN	Stack MERN	9. Seguridad	Security
5. React	React	10. Desarrollo de software	Software development

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este proyecto se hizo con el fin de crear una aplicación web para la identificación de taxis que pueda mostrar información de un vehículo y de conductor a partir de leer un QR. Esta aplicación surge a partir de un problema relacionado con la percepción de inseguridad relacionada con los taxis ya que se cree que existe una brecha entre la veracidad de la información y el usuario, pues la tarjeta que usan los taxistas puede ser muy fácil de falsificar, pero además en muchas ocasiones no se ve esta tarjeta.

La aplicación web se hizo con el stack MERN (MongoDB, Express, React, Node.js) con el fin de que la aplicación pueda ser escalable, flexible de desarrollar, rápida, entre otros.

El objetivo es proporcionar a los usuarios la posibilidad de verificar información de un taxi mediante un código QR, lo que puede contribuir a mejorar la seguridad en el transporte. La aplicación también permitiría a los conductores de taxi o una empresa de taxis pueda registrarse y agregar información sobre su o los vehículos, lo que podría proporcionar un mayor nivel de transparencia y confianza para los usuarios.

ABSTRACT: (Máximo 250 palabras)

This project was created to develop a web application for the identification of taxis that can display information about a vehicle and its driver by scanning a QR code. The application was born out of a problem related to the perceived insecurity associated with taxis, as there is a gap between the accuracy of the information and the user, given that the card used by taxi drivers can be easily falsified, and it is not always visible.

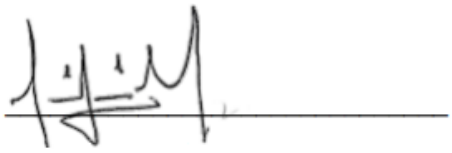
The web application was built using the MERN stack (MongoDB, Express, React, Node.js) to make it scalable, flexible to develop, fast, among other benefits.

The aim is to provide users with the ability to verify information about a taxi through a QR code, which could contribute to improving transportation security. The application also allows taxi drivers or companies to register and add information about their vehicles, which could provide a higher level of transparency and trust for users.

APROBACION DE LA TESIS

Nombre Jurado: Johan Julián Molina Mosquera

Firma:



Nombre Jurado: Martin Diomedez Bravo Obando

Firma:

APLICACIÓN WEB PARA IDENTIFICACIÓN DE TAXIS CON TECNOLOGÍA QR

JOHAN SEBASTIAN GUTIERREZ REPIZO
JUAN PABLO PENAGOS LOSADA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2023

APLICACIÓN WEB PARA IDENTIFICACIÓN DE TAXIS CON TECNOLOGÍA QR

JOHAN SEBASTIAN GUTIERREZ REPIZO cod: 20162153434
JUAN PABLO PENAGOS LOSADA cod: 20172163408

Trabajo de grado para aplicar
al título de ingeniero electrónico

Director:
Mag. Jesús David Quintero Polanco

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2023

Notas de aceptación

_____
_____
_____
_____
_____
_____
_____

_____
Firma del director de Tesis

_____
Firma del Jurado

_____
Firma del Jurado

Neiva, 10 de abril de 2023.

A mis padres, Albeiro Gutierrez y Diana Milena Repizo.

Johan Sebastian Gutierrez Repizo

AGRADECIMIENTOS

Gracias a Dios y a la vida por haberme puesto en este camino. Gracias a mis padres Albeiro y Milena, mis hermanas, Breidy, Salomé y Violeta por todo su apoyo, sacrificio y amor, a mis abuelos, mis tíos, primos y familiares, pero de manera especial a quienes me acogieron en su hogar y me hicieron sentir como en mi casa.

A mi novia, Mariana Rojas por todo su apoyo y amor incondicional en este bonito viaje.
A Monje, Acosta, Marlly y demás amigos que me dejaron la universidad por su amistad.
A la Universidad Surcolombiana por abrirme sus puertas.

Y a cada una de las personas que pasaron en mi vida en este camino por la universidad.

Johan Sebastian Gutierrez Repizo

Quiero aprovechar esta oportunidad para expresar mi profundo agradecimiento a mis hermanos y a mis padres que han sido un gran apoyo para mí en mi vida. A través de los altibajos, siempre me han animado a seguir adelante y perseguir mis metas con determinación. Me han brindado la confianza y el amor incondicional para que pueda alcanzar mis objetivos y superar mis obstáculos. Gracias por estar allí para mí cuando lo necesito y por alentarme a dar lo mejor de mí en todo momento. Aprecio profundamente el sacrificio y el esfuerzo que han hecho para apoyarme.

Juan Pablo Penagos Losada

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

GLOSARIO

Stack: Es un término que se utiliza en desarrollo de software para referirse al conjunto de tecnologías y herramientas que se utilizan en un proyecto

Ride sourcing: Es un término utilizado en la industria del transporte para describir el proceso de conectar a los pasajeros con los conductores a través de una aplicación móvil o una plataforma web.

NoSQL: Es un tipo de base de datos que no utiliza el lenguaje de consultas SQL (Structured Query Language). Las bases de datos NoSQL son muy útiles en aplicaciones que manejan grandes cantidades de datos no estructurados o semiestructurados.

API: Es el acrónimo de "Application Programming Interface". Se refiere a un conjunto de protocolos, reglas y herramientas que se utilizan para interactuar con una aplicación o un servicio en línea. Las APIs permiten que diferentes sistemas puedan comunicarse entre sí de manera eficiente.

CRUD: Es el acrónimo de Create, Read, Update, Delete. Se refiere a las cuatro operaciones básicas que se realizan sobre una base de datos: crear, leer, actualizar y borrar registros.

Access Token: Es un código que se utiliza en autenticación y autorización en línea para identificar a un usuario y permitirle el acceso a recursos específicos.

Modal: Es un componente de interfaz de usuario que se utiliza para mostrar información o solicitar una acción del usuario. Un modal generalmente aparece en la pantalla en una capa superpuesta y oscurece el resto de la interfaz. Los modales son útiles para mostrar información adicional sin interrumpir el flujo de la aplicación principal.

Hosting: El hosting (o alojamiento web) es un servicio que permite almacenar y publicar un sitio web en internet para que sea accesible desde cualquier lugar del mundo.

RESUMEN

Este proyecto se hizo con el fin de crear una aplicación web para la identificación de taxis que pueda mostrar información de un vehículo y de conductor a partir de leer un QR. Esta aplicación surge a partir de un problema relacionado con la percepción de inseguridad relacionada con los taxis ya que se cree que existe una brecha entre la veracidad de la información y el usuario, pues la tarjeta que usan los taxistas puede ser muy fácil de falsificar, pero además en muchas ocasiones no se ve esta tarjeta.

La aplicación web se hizo con el stack MERN (MongoDB, Express, React, Node.js) con el fin de que la aplicación pueda ser escalable, flexible de desarrollar, rápida, entre otros.

El objetivo es proporcionar a los usuarios la posibilidad de verificar información de un taxi mediante un código QR, lo que puede contribuir a mejorar la seguridad en el transporte. La aplicación también permitiría a los conductores de taxi o una empresa de taxis pueda registrarse y agregar información sobre su o los vehículos, lo que podría proporcionar un mayor nivel de transparencia y confianza para los usuarios.

ABSTRACT

This project was created to develop a web application for the identification of taxis that can display information about a vehicle and its driver by scanning a QR code. The application was born out of a problem related to the perceived insecurity associated with taxis, as there is a gap between the accuracy of the information and the user, given that the card used by taxi drivers can be easily falsified, and it is not always visible.

The web application was built using the MERN stack (MongoDB, Express, React, Node.js) to make it scalable, flexible to develop, fast, among other benefits.

The aim is to provide users with the ability to verify information about a taxi through a QR code, which could contribute to improving transportation security. The application also allows taxi drivers or companies to register and add information about their vehicles, which could provide a higher level of transparency and trust for users.

INTRODUCCIÓN

En los últimos años el factor seguridad se ha vuelto en ocasiones definitivo a la hora de elegir un servicio que pueda llevar de un lugar a otro dentro de la ciudad. Esto debido a los constantes hechos de violencia que se presentan de manera recurrente en el servicio público. Empresas ride sourcing a nivel mundial y nacional como los son Uber, InDriver, Didi, etc., Han tenido gran éxito y este es debido al desarrollo de estrategias que atraen al usuario como lo son la fiabilidad de información respecto a la ruta, el conductor y el precio del servicio, y así incentivar la recomendación del servicio y la recompra del mismo. (Enrique et al., 2021)

En el capítulo IV del decreto 172 de 2001 se reglamenta la tarjeta de control. La tarjeta de control es el documento legal que todo taxi debe poseer y que debe contener información de él y de su conductor. Aunque hay una reglamentación que obliga a las empresas de taxis a tener una tarjeta de control, no hay un instrumento para que el usuario pueda verificar la veracidad de la información que está ofrece. En este capítulo del decreto también se menciona que la tarjeta debe de estar en la parte trasera del asiento del copiloto. Es por esto por lo que un usuario de taxi no puede hacer la verificación de la legalidad de un taxi si no hasta que se está dentro de este.

Usar un código QR para la identificación de un taxi, donde se pueda acceder a información de manera rápida en una página web con tal de hacer la verificación de algunos datos relacionados con el taxi y el conductor, puede ayudar al usuario de taxi de que se cometan contra él actos que atenten contra su integridad.

El stack MERN es uno de los más usados en el mundo de la programación, usa como base el lenguaje JavaScript y también tecnologías como lo son MongoDb, ExpressJS, React y NodeJS. Se pretende que con este estudio se pueda dar una solución a los taxistas y usuarios de taxis en relación con la identificación de taxis seguros.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Realizar una aplicación web para que a partir de la lectura de un código QR muestre información relacionada con un taxi y su conductor que sirva como una posible herramienta para disminuir la percepción de inseguridad.

1.2 OBJETIVOS ESPECÍFICOS

- Determinar qué información básica relacionada con el taxi y su conductor puede ir en la página web.
- Diseñar el frontend y backend de la aplicación web.
- Elegir el estilo y la forma en la que se van a crear los códigos QR.
- Validar el funcionamiento del aplicativo web.

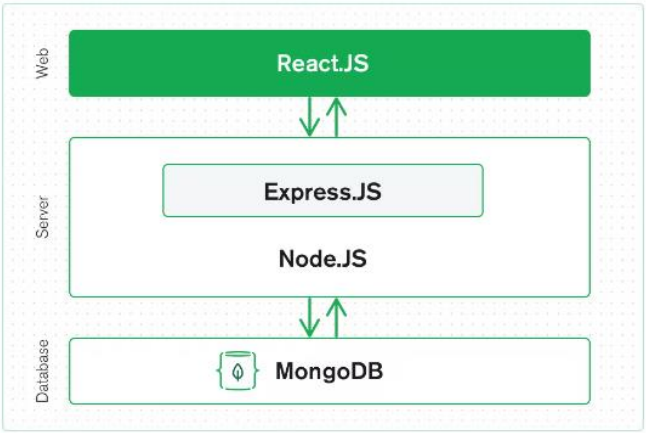
2. FUNDAMENTOS BÁSICOS

2.1 MERN STACK

Un Stack de tecnologías es una combinación de lenguajes de programación, frameworks, bibliotecas, patrones y diseños de UI/UX utilizados por los desarrolladores para implementar un sistema o aplicación (MIgliorata, et al.,2020).

MERN es un acrónimo que se refiere a las tecnologías que se utilizan para construir aplicaciones web modernas y escalables:

- MongoDB: una base de datos NoSQL altamente escalable y flexible que utiliza documentos JSON para almacenar datos. Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es
- necesario seguir un esquema (Hernández Gonzales,2020).
- Express: un marco de aplicaciones web para Node.js que proporciona una amplia gama de características, incluyendo el enrutamiento, gestión de errores y soporte para middleware. Es utilizado para crear y manejar las rutas y controladores que se encargan de interactuar con la base de datos MongoDB, así como para la autenticación y autorización de los usuarios.
- React: una biblioteca de JavaScript de código abierto para crear interfaces de usuario modernas y dinámicas de una sola página.
- Node.js: un entorno de tiempo de ejecución de JavaScript de código abierto que permite a los desarrolladores crear aplicaciones web altamente escalables y de alto rendimiento utilizando JavaScript tanto en el servidor como en el cliente.



Fuente: MERN Stack Explained https://www.mongodb.com/mern-stack

Figura 1. Arquitectura MERN

2.1.1 Mongoose

Mongoose es una herramienta de modelado de objetos de MongoDB diseñada para trabajar en un entorno de Node.js. Permite definir modelos de datos en un esquema de Mongoose para MongoDB, lo que simplifica la integración con Node.js y proporciona una forma estructurada de interactuar con la base de datos.es una herramienta esencial para trabajar con MongoDB en un entorno de Node.js. Simplifica el modelado y la manipulación de datos en la base de datos, proporciona una estructura clara y definida para interactuar con los datos, y agrega una serie de características útiles para trabajar con MongoDB.

2.1.2 JSONWebToken

Jsonwebtoken es una librería de JavaScript que es utilizada para generar y verificar tokens de autenticación en aplicaciones web y móviles. Los JWT son un estándar abierto (RFC 7519) que se utilizan para transmitir información de forma segura entre diferentes partes de una aplicación.

Jsonwebtoken permite crear tokens de autenticación seguros que pueden ser compartidos entre diferentes partes de una aplicación y se pueden utilizar para autorizar solicitudes.

Funciona mediante la creación de un token que contiene información del usuario, como su identificación y cualquier permiso o rol que tenga dentro de la aplicación. Este token es entonces firmado digitalmente utilizando una clave privada, lo que garantiza que la información del usuario no puede ser manipulada por terceros.

Cuando un usuario envía una solicitud a una aplicación, se incluye el token en la solicitud. La aplicación utiliza la biblioteca jsonwebtoken para verificar el token y autenticar al usuario. Si el token es válido, el usuario tiene acceso a los recursos solicitados. Si no es válido, se niega el acceso. Todos estos procesos tienen un nivel de encriptación que aumenta el nivel de seguridad de la aplicación, desde el lado del cliente se usa una librería llamada “jwt-decode” que se encarga de hacer la decodificación del token.

2.1.3 BCRYPTJS

Es una librería de encriptación que utiliza funciones hashing para encriptar contraseñas. Lleva incorporado un valor llamado salt, que es un fragmento aleatorio que se usará para generar el hash asociado a la password (Martinez Garcia, 2020).

Cuando un usuario crea una cuenta en una aplicación web, debe proporcionar una contraseña. En lugar de almacenar esta contraseña en texto plano, se utiliza un algoritmo de hashing como Bcrypt para cifrar la contraseña antes de almacenarla en la base de datos. Esto significa que la contraseña no se puede leer directamente desde la base de datos y, por lo tanto, se vuelve mucho más difícil para un atacante recuperar la contraseña original.

2.1.4 Semantic UI React

Semantic UI React es una librería de componentes de interfaz de usuario diseñada para React, que se basa en Semantic UI. Proporciona una amplia gama de componentes reutilizables predefinidos, como botones, menús, formularios, tablas y entre otros, pueden ser utilizados en la construcción de interfaces de usuario para aplicaciones web.

Es una librería que tiene un estilo visual moderno y limpio, su enfoque está en la usabilidad y la accesibilidad. También incluye herramientas de personalización y temas predefinidos para permitir a los desarrolladores adaptar la apariencia de sus aplicaciones a sus necesidades.

Semantic UI React es una opción popular para proyectos de React debido a su facilidad de uso, documentación completa y comunidad activa. Además, también se integra bien con otras herramientas y bibliotecas de JavaScript, como Redux, React Router y Next.js.

2.1.5 React QR

React QR es una librería de componentes de React que permite generar y leer códigos QR en aplicaciones web. Proporciona una forma fácil de integrar códigos QR en una aplicación React, lo que permite a los usuarios escanear y compartir información de manera rápida y conveniente. Los componentes de React QR están diseñados para ser personalizables y se pueden adaptar fácilmente para cumplir con las necesidades específicas de cada proyecto. Además, React QR también proporciona una serie de opciones para el diseño y estilo de los códigos QR generados, lo que permite una mayor flexibilidad y control en su uso. Usar esta librería tiene ventajas como lo son la integración sencilla, personalización, flexibilidad, compatibilidad, rapidez, entre otros.

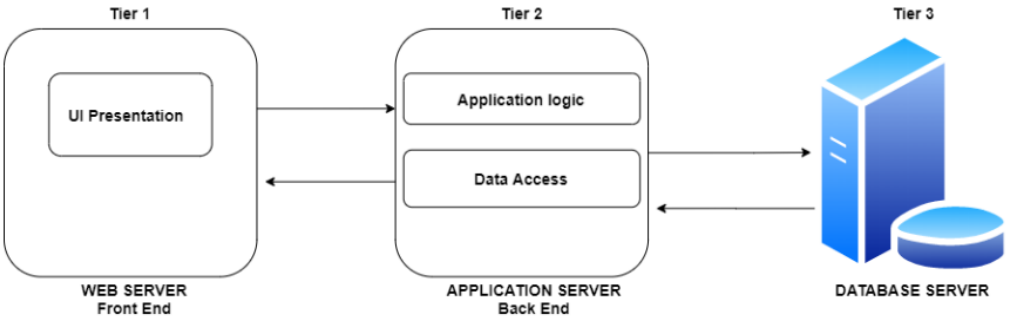
2.1.6 Códigos QR

Los códigos QR son una manera fácil y rápida de acceder a la información. Un código QR es un código de dos dimensiones compuesto por pequeños cuadros negros dentro de uno más grande que generalmente es blanco. Hoy en día la mayoría de los teléfonos celulares inteligentes están equipados con aplicaciones propias para hacer lectura de los códigos QR, este lector interpreta los datos que están en código y redirige hacia la información que el creador del código ha realizado. (J, H. 2022).

Los códigos QR se pueden poner casi que en cualquier parte y sus usos son muy variados, pues van desde ayudar a hacer un pago, hasta iniciar sesión en algún sitio web.
Los QR tienen ventajas como que se pueden leer sin necesidad de un aparato específico y que generalmente son más duraderos, pues la información puede ser decodificada aún si alguna parte del código está dañada, también pueden ser leídos en cualquier dirección.
Uno de los valores más importantes de los QR es que pueden conectar a una persona con cualquier información de manera rápida, pero también que la información que se puede poner “detrás” de un QR está limitada por la imaginación, lo que abre un infinito mundo de posibilidades.

3. CREACIÓN DE LA APLICACIÓN WEB

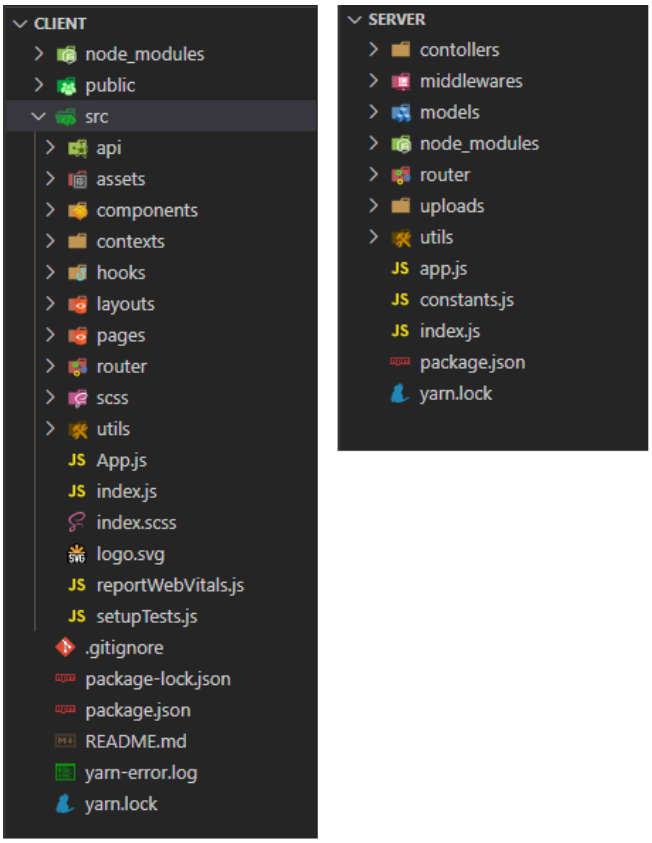
La aplicación web se crea bajo una arquitectura 3 capas (3-tier en inglés) (Figura 2), de tal manera que cada una de las capas puede ser desarrolladas de manera simultánea e independiente permitiendo así que pueda actualizar y/o escalar de manera más fácil.



Fuente: Elaboración propia

Figura 2. Arquitectura de la aplicación web

Además, se tienen en cuenta los conceptos de adhesión y acoplamiento, los cuales garantizan que teniendo un bajo acoplamiento y una alta cohesión se mejore la mantenibilidad de los módulos de la aplicación, la reutilización de componentes, etc.



Fuente: Elaboración propia

Figura 3. Módulos y componentes de la aplicación web.

A partir de las tecnologías anteriormente mencionadas se crea una aplicación web para la identificación de taxis con QR.

Teniendo en cuenta el uso que se le va a dar a la aplicación, los autores han decidido llamarla “Qrtaxi”.

3.1 HOME, LOGIN Y REGISTRO

La página de Home es a la que se dirige por defecto cuando se hace una petición a la URL principal de la aplicación web. Esta página está compuesta por un footer, donde hay información relacionada al proyecto, también hay una imagen y un botón para ingresar a la página de login y registro.

Fuente: Elaboración propia

Figura 4. Página de inicio (Home) de la aplicación.

La página de Login y Registro se crea usando el framework semantic ui de react para la parte del front. El manejo y validación del formulario del login y registro se hace con las librerías formik y yup, estos nos permiten validar que se cumplan con un esquema de registro previamente establecidos.

```
import * as Yup from "yup";

export function initialValues () {
    return{
        email:"",
        password:"",
        repeatPassword:"",
        conditionsAccepted: false,
    };
};

export function validationSchema() {
    return Yup.object({
        email: Yup.string().email("Email no valido").required("Campo obligatorio"),
        password: Yup.string().required("Campo obligatorio").min(8,"La contraseña debe tener mínimo 8
        caracteres"),
        repeatPassword: Yup.string().required("Campo obligatorio").oneOf([Yup.ref("password")],"Las
        contraseñas deben de ser iguales"),
        conditionsAccepted: Yup.bool().isTrue(true)
    })
}
```

Fuente: Elaboración propia

Figura 5. Fragmento de código de esquema para registro de usuario.

Para la verificación de los usuarios que usan la aplicación y la protección de los recursos expuestos por API, dependiendo del endpoint se usa una autenticación única por medio de tokens para cada uno de los usuarios que acceden a la página. Se usa JSON Web Token que es un estándar abierto para la creación de tokens de acceso que ayuda a manejar los privilegios en la aplicación.

En cuanto a la protección de contraseñas se hace un cifrado para guardarla y una decodificación para iniciar sesión usando la librería bcrypt para NodeJS, la cual después de haberse importado, se ejecuta haciendo que la contraseña se guarde y se use de forma segura (Figura 6). La función “genSaltSync()” permite que se genere de manera síncrona una cadena de caracteres de manera aleatoria, haciendo que el hash de cifrado sea único aún si las contraseñas son iguales.

La petición que se hace a la base de datos para el registro y el login es de tipo POST a la rutas de la API “/auth/register” y “/auth/login” respectivamente.

```
const bcrypt = require ("bcryptjs")
const User = require("../models/user.js")
const jwt = require("../utils/jwt")

function register (req, res){
    const{firstname,lastname,password,email} = req.body;
    const user = new User({...
    });
    const salt = bcrypt.genSaltSync(10);
    const hashPassword = bcrypt.hashSync(password,salt)
    user.password = hashPassword;

    user.save((error,userStorage)=>{
       ...
    })

}
```

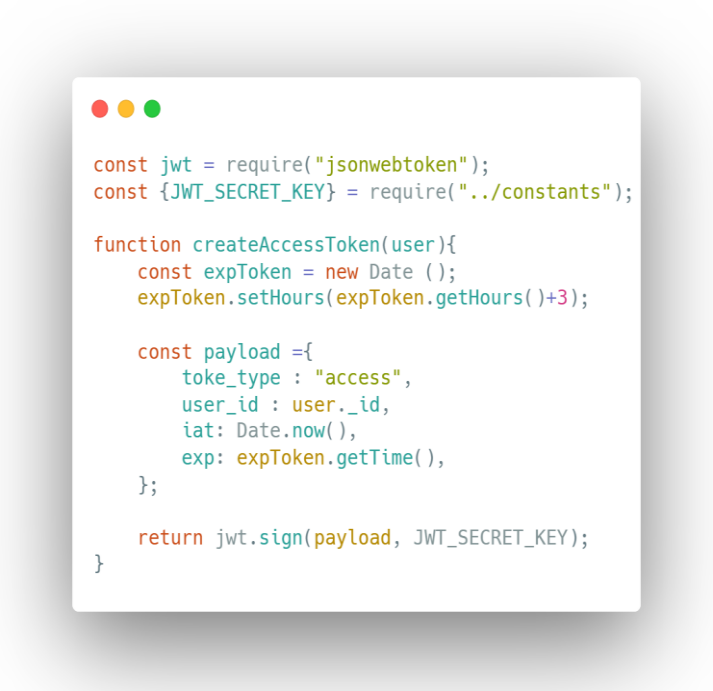
Fuente: Elaboración propia

Figura 6. Fragmento de código del controlador de contraseñas para uso de bcrypt en el registro.

Fuente: Elaboración propia

Figura 7. Página para el formulario de Login e inicio de sesión.

Una vez se inicia sesión, se crea un token para el usuario (Figura 8) el cual es usado para autenticar y/o autorizar al usuario a hacer ciertas peticiones dentro de la página. Para este caso el token tendrá una fecha de expiración de 3 horas a partir del inicio de sesión. Cada usuario tiene asignado un payload que contiene los datos que se incluyen en el token. El middleware de autenticación de NodeJS (Figura 9) es donde se usa el token.



```
const jwt = require("jsonwebtoken");
const {JWT_SECRET_KEY} = require("../constants");

function createAccessToken(user){
    const expToken = new Date ();
    expToken.setHours(expToken.getHours()+3);

    const payload ={
        toke_type : "access",
        user_id : user._id,
        iat: Date.now(),
        exp: expToken.getTime(),
    };

    return jwt.sign(payload, JWT_SECRET_KEY);
}
```

Fuente: Elaboración propia

Figura 8. Fragmento de código donde se crea el Access Token.

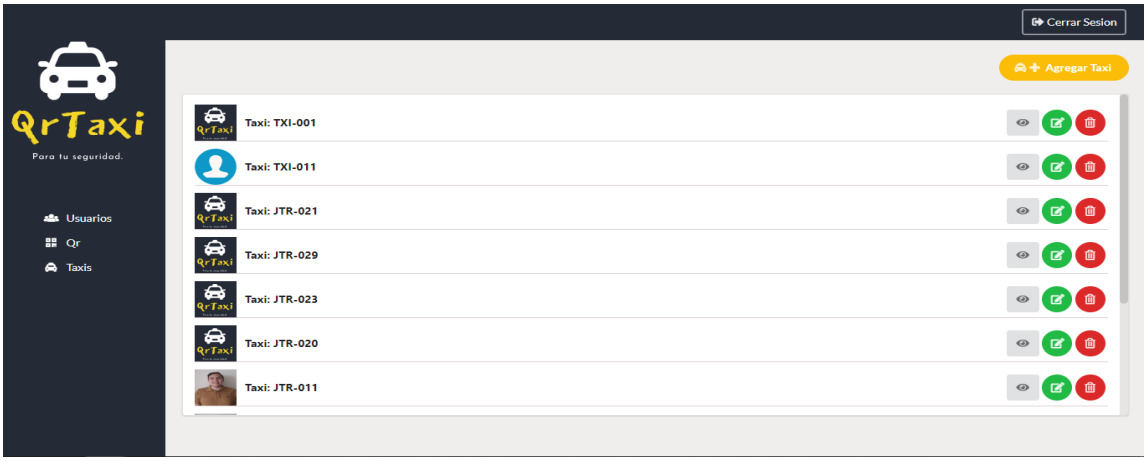
```
const jwt = require("../utils/jwt");

function asureAuth(req, res, next) {

    if(!req.headers.authorization){...
          }
    const token = req.headers.authorization.replace("Bearer ","");

try {
    const payload = jwt.decoded(token);
    const {exp}= payload;
    const currentDate = new Date().getTime();

    if(exp <= currentDate){
       ...
    }
    req.user = payload;
    next();

} catch (error) {...
}


};
```

Fuente: Elaboración propia

Figura 9. Middleware de autenticación

3.2. PÁGINA DE ADMINISTRACIÓN

Una vez se obtiene el usuario y se le asigna el token, la página que se muestra es la página de administración en la sección de gestión de taxis (Figura 10).



Fuente: Elaboración propia

Figura 10. Página de administración

La página de administración tiene un layout para el menú y otro para mostrar la administración de los usuarios, los taxis y generación del QR.

3.2.1 ADMINMENU

El “AdminMenu” es un componente (Figura 11) que es hecho a partir de la librería “Menú” de semantic que genera un menú vertical. Se muestra a la izquierda de la figura 10. Desde aquí, gracias a React Router, se puede hacer navegación a las diferentes rutas de administración. Cuando se dé clic en alguno de los iconos del menú, esté se quedará resaltado para que el usuario sepa en qué ruta está.

```
return (
    <Menu fluid vertical icon text className='admin-menu'>
        <Menu.Item as={Link} to ="/admin/users" active={isCurrentPath("/admin/users")}>
            <Icon name='users'/>
            Usuarios
        </Menu.Item>

        <Menu.Item as={Link} to ="/admin/qr" active={isCurrentPath("/admin/menu")}>
            <Icon name='qrcode'/>
            Qr
        </Menu.Item>

        <Menu.Item as={Link} to ="/admin/taxis" active={isCurrentPath("/admin/taxis")}>
            <Icon name='taxi'/>
            Taxis
        </Menu.Item>
    </Menu>
  )
```

Fuente: Elaboración propia

Figura 11. Fragmento de código del componente “AdminMenu”

3.2.1.1 Usuarios

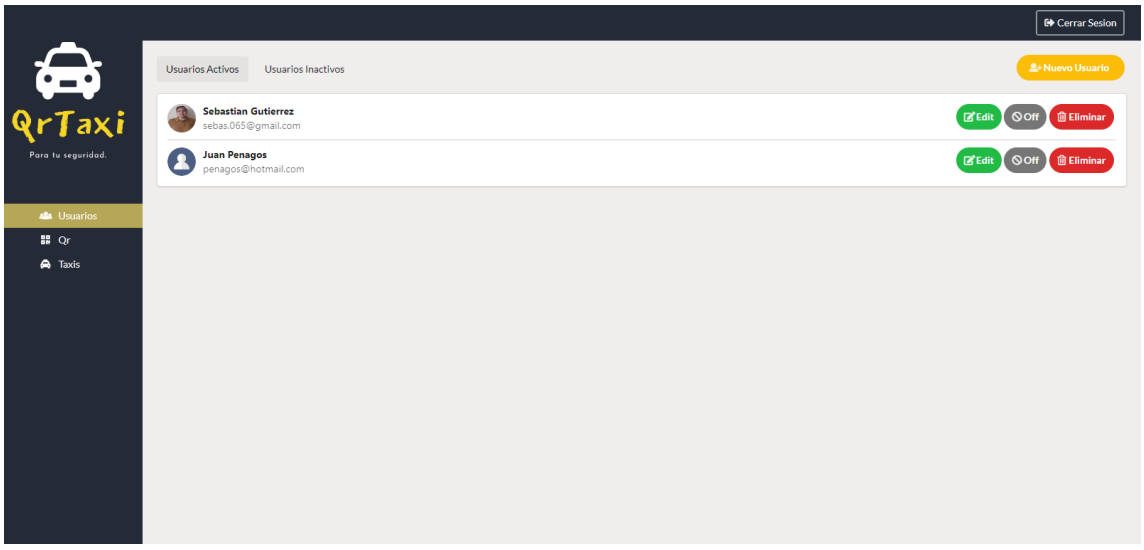
Para la creación de usuarios se hace un esquema en la API para que MongoDB pueda guardar los datos en el momento que el usuario lo requiera.

```
const mongoose = require("mongoose");
const UserSchema = mongoose.Schema({
    firstname:String,
    lastname:String,
    email: {
        type: String,
        unique: true,
    },
    password:String,
    role:String,
    active:Boolean,
    avatar:String,
});
module.exports = mongoose.model("User", UserSchema);
```

Fuente: Elaboración propia

Figura 12. Fragmento de código esquema de usuarios.

La propiedad “email” del esquema hace referencia a que no se pueden crear dos usuarios con correos iguales. Este modelo puede ser utilizado por una aplicación para crear, leer, actualizar y eliminar usuarios en la base de datos MongoDB, utilizando las funciones de Mongoose y MongoDB.



Fuente: Elaboración propia

Figura 13. Página de administración de usuarios.

Se usa un componente de React para mostrar la página de administración de usuarios. Al cargar, se muestran dos pestañas en la parte superior, una para "Usuarios Activos" y otra para "Usuarios Inactivos". En cada pestaña se muestra una lista de usuarios correspondiente, ya sea activos o inactivos.

El componente también incluye un botón "Nuevo Usuario" que permite abrir un modal donde se puede crear un nuevo usuario utilizando un formulario de entrada. Este formulario se muestra dentro de un componente “BasicModal”, que se importa desde otro archivo. Desde esta página se puede hacer una operación completa de CRUD (Create, Read, Update, Delete en inglés) para los usuarios.

Además, el componente utiliza el paquete Semantic UI React para construir los elementos de la interfaz de usuario, como los botones, los iconos y las pestañas. Es importante mencionar que se usa un componente “pane” de Semantic (Figura 14) para crear la sección de usuarios activos e inactivos, pues gracias a este componente se puede presentar el contenido de una manera estructurada y que es visualmente atractiva y en diferentes pestañas.

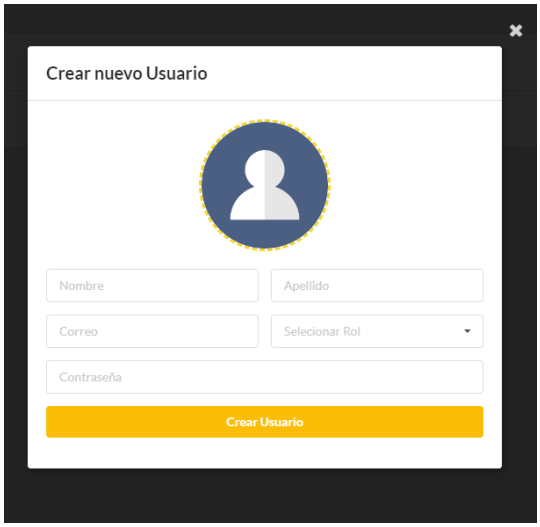
```
const panes =[
   {
     menuItem: "Usuarios Activos",
     render: () => (
       <Tab.Pane attached = {false}>
         <ListUser usersActive = {true} reload={reload} onReload={onReload}/>
       </Tab.Pane>
     )
   },
   {
     menuItem: "Usuarios Inactivos",
     render: () => (
       <Tab.Pane attached = {false}>
         <ListUser usersActive = {false} reload={reload} onReload={onReload} />
       </Tab.Pane>
     )
   }

 ]
```

Fuente: Elaboración propia

Figura 14. Fragmento de código del componente “panes”

Cuando se hace clic en nuevo usuario, se abre un modal (Figura 15) que muestra un formulario donde se puede crear un nuevo usuario teniendo en cuenta el esquema que anteriormente se mencionó. Igual que en los formularios de Login y

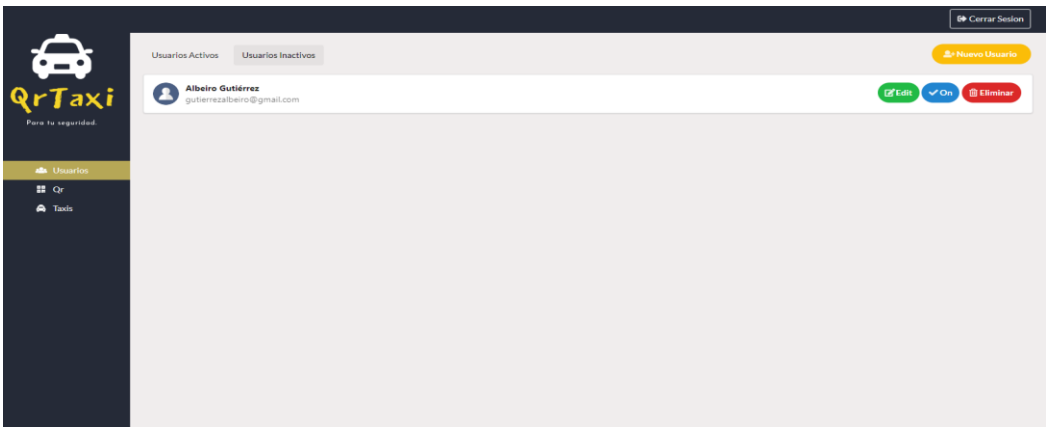
registro, se hace una validación y manejo de los datos suministrados por el usuario con la librería “Yup” y “Formik” de react. Este mismo modal, es usado para la actualización de datos de usuario, sin embargo, cuando se va a actualizar un usuario aparece en cada uno de los campos la información que el usuario tiene. Es por eso que se crea un componente modal, para que este se pueda renderizar según el uso.



Fuente: Elaboración propia

Figura 15. Modal para creación de nuevo usuario.

Aunque un usuario se registre, este no puede acceder hasta que un usuario que ya tiene permisos lo active desde su perfil en la aplicación. Además, se podrá hacer la operación CRUD para los usuarios inactivos desde esta pestaña. Un usuario que esté inactivo puede volverse activo y viceversa en cualquier momento.



Fuente: Elaboración propia

Figura 16. Pestaña de usuarios inactivos.

Las peticiones del CRUD y de activar o desactivar un usuario, son peticiones que están siempre autenticadas con el “accesToken” que se genera al iniciar sesión.

```
async deleteUser(accessToken,idUser){
        try {
            const url = `${this.baseApi}/${ENV.API_ROUTES.USER}/${idUser}`
            const params = {
                method: "DELETE",
                headers:{
                    Authorization: `Bearer ${accessToken}`,
                }
            };

            const response = await fetch (url,params)
            const result = await response.json()
            if(response.status !== 200) throw result;
            return result;

        } catch (error) {
            throw error;
        }
    }
```

Fuente: Elaboración propia

Figura 17. Fragmento de código para eliminación de usuario.

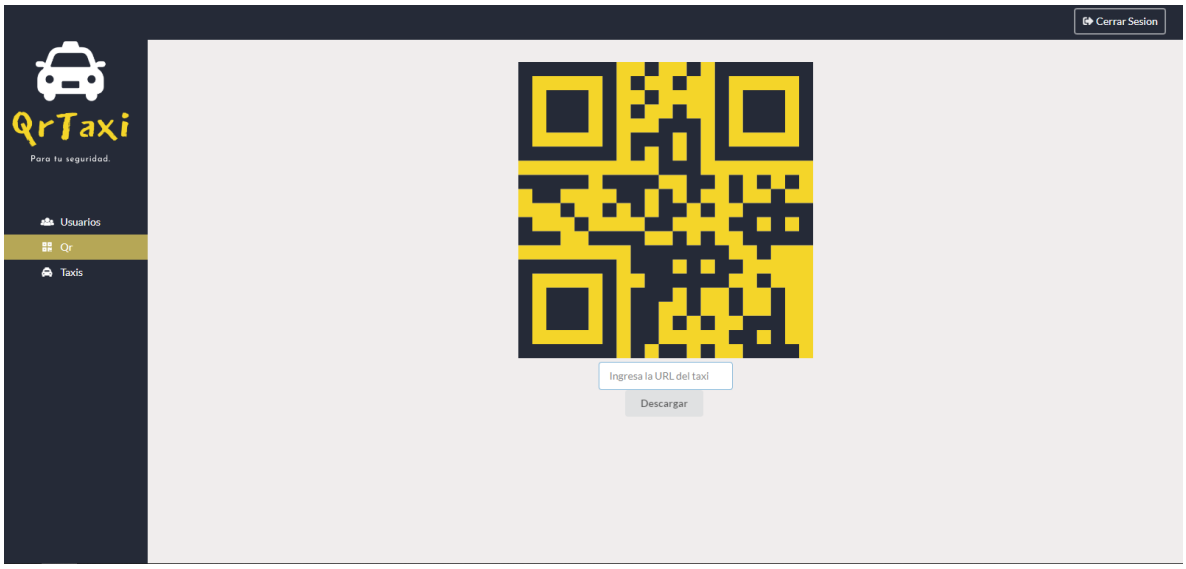
3.2.1.2 QR

Desde esta sección el usuario puede generar un código QR el cual podrá descargar para que después se pueda poner en el taxi.

Para hacer este componente se utiliza la librería “react-qr-code” la cual renderiza un código QR a partir de una entrada. Esta librería permite darle estilos al QR tales como color de fondo, tamaño, etc.

Cada código QR puede ser descargado en un formato PNG. Esto se hace mediante la creación de un elemento de imagen en el DOM, dibujando el código QR en un canvas y luego convirtiendo el canvas en una imagen PNG que se descarga automáticamente. El manejo del evento onClick del botón de descarga llama a la función “donwloadQr”, que contiene el código necesario para realizar esta tarea.

El uso de la “react-qr-code” hace que esta sección de la aplicación web sea bastante sencilla y de fácil implementación.



Fuente: Elaboración propia

Figura 18. Página de creación y descarga de QR.

```
return (
   <div className='Qr'>

   <div className='QrImage' >
       <QRCode
       size={400}
       bgColor= '#F4D529'
       fgColor= "#252A37"
       level='L'
       value={inputValue}
       id="QRCode"
       />
   </div>

   <Input  focus placeholder="Ingresa la URL del taxi" onChange={(e)=>setInputValue(e.target.value)}>
     </Input>

   <Button animated onClick={donwloadQr} value="Download">
   <Button.Content visible> Descargar </Button.Content>
   <Button.Content hidden>
   <Icon name='download' />
   </Button.Content>
   </Button>

   </div>
 )
```

Fuente: Elaboración propia

Figura 19. Fragmento de código de creación y descarga del código QR.

3.2.1.3 Taxis

De la misma manera en que se hizo con los usuarios, Taxis tiene un esquema **(Figura 20)** de Mongoose para el modelo de un taxi: Aquí los datos de placa serán de valor único, para evitar que se registren taxis con placas iguales.

```
const mongoose = require("mongoose");
const mongoosePaginate = require("mongoose-paginate");

const TaxiSchema =mongoose.Schema({
    placa: {
        type: String,
        unique: true,

    },
    empresa : String,
    numeroInterno:String,
    conductor: String,
    activo : Boolean,
    foto : String,
    path: {
        type:String,
        unique: true,
    }
});

TaxiSchema.plugin(mongoosePaginate);

module.exports = mongoose.model("Taxi",TaxiSchema);
```

Fuente: Elaboración propia

Figura 20. Fragmento de código esquema de taxi.

Desde esta parte de la aplicación web se puede hacer la administración completa de cada uno de los taxis. Al igual que en la sección de usuarios, los taxis se listan para ser mostrados en un componente. En la API se crea un controlador (Figura 21) para manejar cada una de las peticiones de CRUD que se hace en la sección taxi del front.

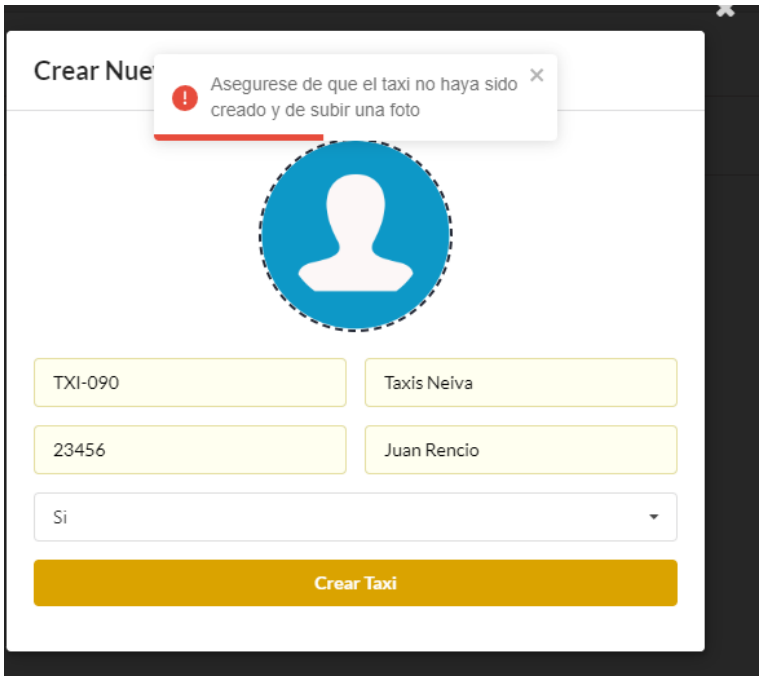
```
async function createTaxi(req,res){
    const taxi = new Taxi(req.body);

    if(req.files.foto){
        const imagePath = image.getFilePath(req.files.foto);
        taxi.foto = imagePath;
        taxi.path = `QrTaxi-${imagePath.slice(6,14)}`;
    };

    taxi.save((error,taxiStored)=>{
        if(error){
            res.status(400).send({msg:"Error"})
        } else{
            res.status(200).send(taxiStored)
        }
    });

}
```

Fuente: Elaboración propia

Figura 21. Fragmento de código del controlador taxi.

Como ya se había creado un “Modal” este será renderizado y adaptado para la creación y actualización de un nuevo taxi.

A diferencia de la sección usuarios, aquí si no se carga una imagen del conductor del taxi, este no se puede crear, no es opcional. Se harán validaciones de que la placa no haya sido creada y de que el usuario suba una foto del conductor. Se le notificará al usuario con un toast de error para que verifique.

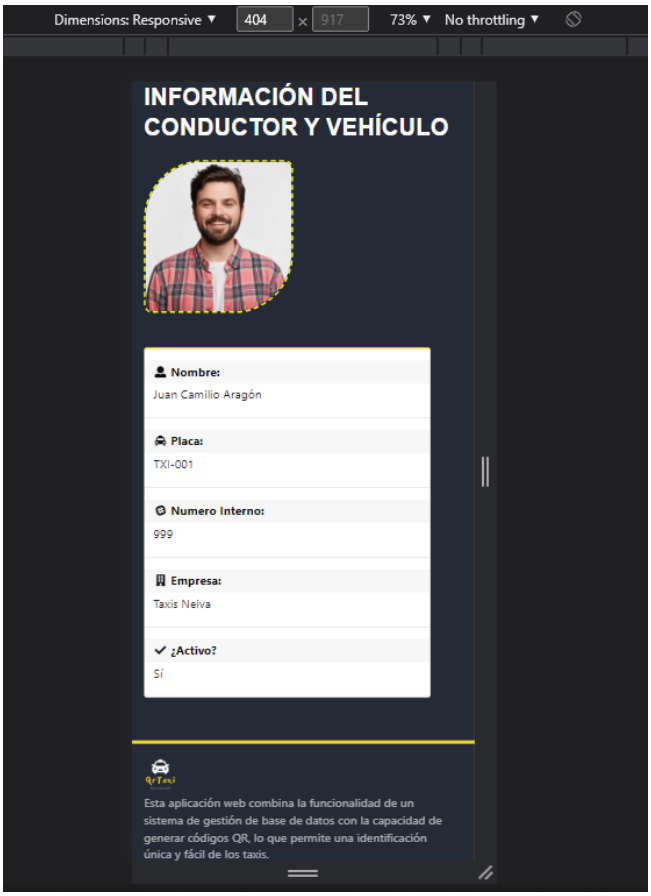


Fuente: Elaboración propia

Figura 22. Toast de error al crear taxi.

En la API se crea un archivo de utilidad que usa una función llamada “getFilePath”, la cual toma un archivo como argumento, que en este caso es la foto de perfil, y devuelve una cadena que representa la ruta de la foto. Como la función crea una cadena de caracteres irrepetible y única para cada archivo, se quitan las “//” que genera el path y se usa para crear un path que será único para cada taxi y que servirá para crear la información guardada del taxi en una página que podrá ver el usuario final.

Se usa un componente “Icono” de Semantic para crear un enlace que llevará al usuario a la información del taxi. La ruta de enlace por defecto será “`/taxi/${taxi.path}`”. Esta ruta es la que se ingresa en el generador de QR para que el usuario del taxi pueda verificar la información del taxi al que se está subiendo. Esta parte de la aplicación web tiene especial detalle en el diseño responsive ya que se espera que siempre se hagan peticiones desde un teléfono celular.



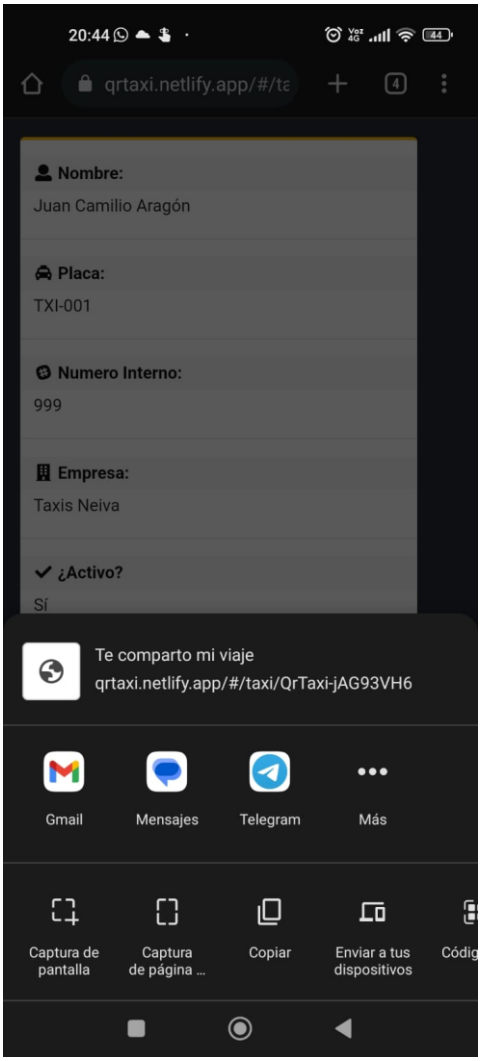
Fuente: Elaboración propia

Figura 23. Diseño responsive de la aplicación web.

Como se había mencionado, la información en esta página se mostrará a partir de la ruta. Se utiliza un estado (useState) para almacenar la información del taxi. Se hace una petición al servidor para obtener los datos del taxi usando el identificador de ruta (path) a través de “TaxiController” que es el controlador del taxi. La información del taxi se almacena en el estado del componente usando la función setTaxi. Si el taxi no está disponible (es decir, aún no se ha obtenido la información del servidor), se muestra un componente “Loader" de Semantic. Una vez obtenida la información del taxi, se muestra la información del conductor y del vehículo en una tabla usando el componente Table de Semantic UI React. La información incluye el nombre del conductor, la placa del vehículo, el número interno del vehículo, la empresa a la que pertenece el vehículo y si el vehículo está activo o no.

La pagina que tiene toda la información del usuario tiene un botón para compartir esta información. Este botón usa una API nativa de javascript llamada “navigator.share” lo que permite compartir contenido de una página web directamente desde el navegador, sin necesidad de copiar y pegar enlaces o utilizar

aplicaciones de terceros, esta API funciona en la mayoría de navegadores modernos, incluyendo Google Chrome, Mozilla Firefox, Microsoft Edge y Safari, entre otros.



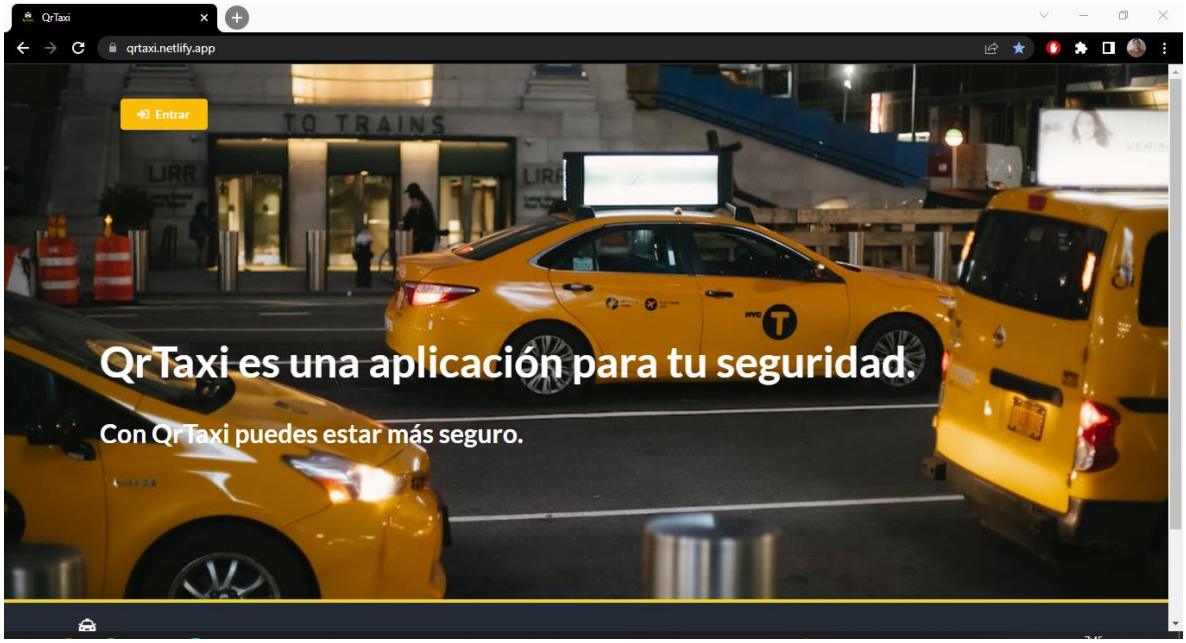
Fuente: Elaboración propia

Figura 24. Opción de compartir viaje.

4. DESPLIEGUE DE LA APLICACIÓN WEB.

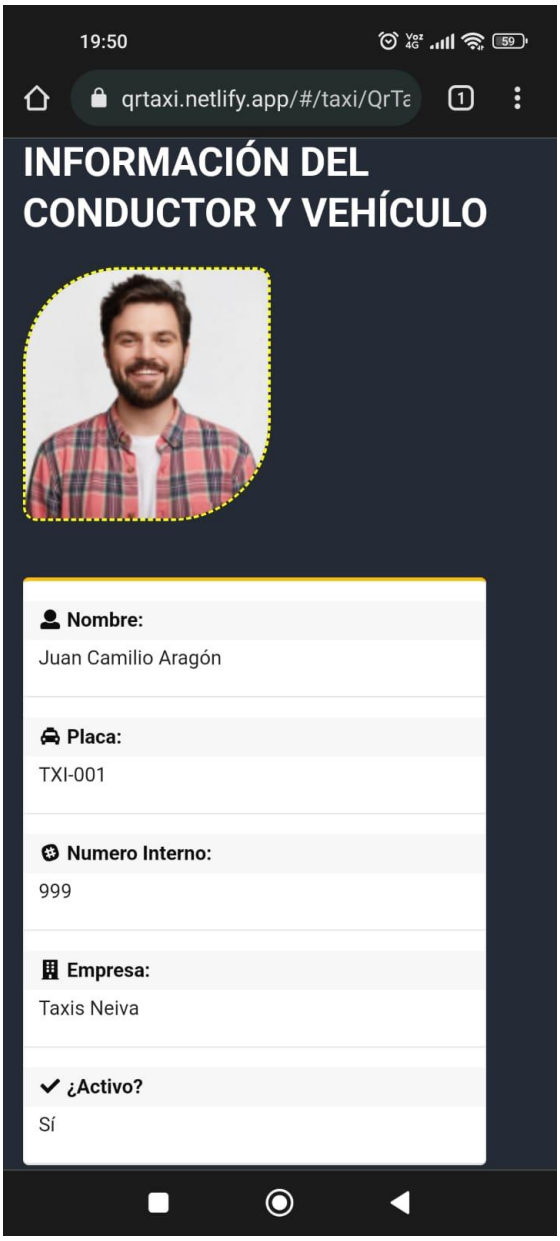
El despliegue de la aplicación se hizo en dos plataformas de hosting como lo son Railway y Netlify es su versión gratuita, que está más que bien para evaluar las funcionalidades y rapidez de la aplicación una vez se les ha hecho el despliegue.

El servidor está alojado en Railway y la parte del cliente está alojado en Netlify. Ambos despliegues se hicieron con un mayor grado de facilidad teniendo todos los códigos alojados en GitHub. GitHub fue también la plataforma de versionamientos que se empleó para hacer el desarrollo de la app.



Fuente: Elaboración propia

Figura 25. Vista de aplicación web en escritorio.



Fuente: Elaboración propia

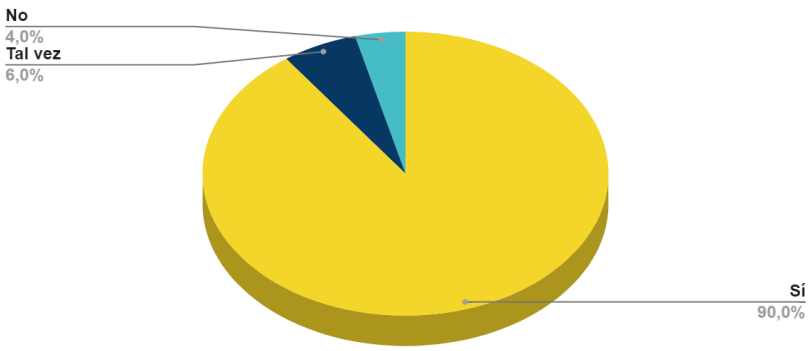
Figura 26. Vista de aplicación web en móvil.

5. ENCUESTA DE ACEPTABILIDAD DE LA APLICACIÓN WEB.

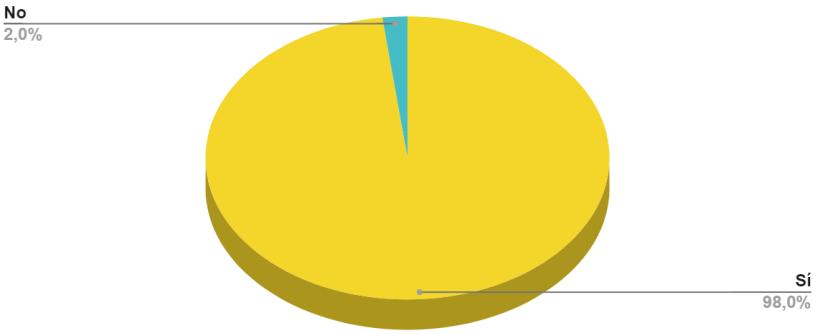
Se realizó una encuesta para conocer la aceptabilidad de la aplicación. A las personas se les explicó el funcionamiento y el contexto del por qué se desarrolló esta aplicación web.

La encuesta se realizó a 100 personas, de diferentes edades y género de la ciudad de Neiva, Aipe y Pitalito. La encuesta se hizo en Google Form y los resultados fueron los siguientes:

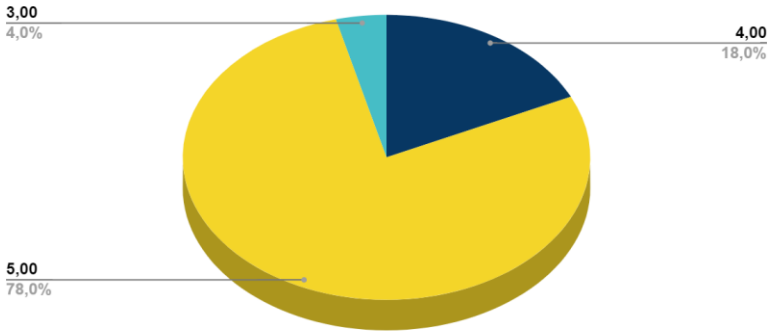
3) ¿Te gustaría ver comentarios o reseñas de otros usuarios sobre el taxi en la página ?



4) ¿Tu percepción de seguridad cambia al ver la información del taxi y del conductor antes de subirte ?



5) En escala de 1 a 5, califica la aplicación.



6. RECOMENDACIONES Y TRABAJOS FUTUROS

Teniendo en cuenta la encuesta que se realizó hay que tener en cuenta que una funcionalidad que se le puede agregar es la de tener reseñas del taxi y del conductor.

Para la parte de inicio de sesión se podría delegar esta parte a software de terceros como lo son “Google”, “auth0” y demás, ya que hacer sistemas de autenticación para aplicaciones más grandes puede ser complicado y además la aplicación puede volverse vulnerable a ataques cibernéticos.

Además, se podría indagar un poco más acerca de qué otra información se le podría agregar relacionada al taxi y el conductor.

Se podría agregar una funcionalidad que haga que los administradores de cada una de las empresas de taxis puedan tener una estadística relacionada con las veces que las personas han tomado u taxi, esta se podría asociar a un botón adicional que el usuario pueda presionar para indicar que tomó el taxi.

En este trabajo no se encuestaron taxistas, pero se recomienda para un futuro trabajo hacerlo, ya que se puede tener una perspectiva importante relacionada con la problemática.

Para hacer un despliegue de la aplicación donde el uso sea masivo, es recomendable comprar dominios y tener en cuenta el precio de los servidores y de las bases de datos.

CONCLUSIONES

La aplicación web que se expuso, surgió como alternativa de solución ante la percepción de inseguridad al momento de usar un taxi. La aplicación web se realizó con tecnologías que se usan en la actualidad en el mundo del desarrollo de software y que tienen amplia documentación lo que facilita la creación de cualquier solución con la tecnología. Se obtuvieron las siguientes conclusiones.

La aplicación web permite que al leer un código QR se acceda a información de un taxi y de su conductor de manera fiable y segura.

La encuesta de aceptabilidad indica que existe una demanda para este tipo de aplicación. Los encuestados están interesados en utilizar una herramienta como esta para verificar la información de los taxis y conductores antes de abordarlos. Se podrían agregar algunas funcionalidades como la de reseñas.

Según los encuestados si se tuviera esta opción, la percepción de seguridad cambiaría, esto debido a que se podría tener una información verificada y confiable del taxi y de su conductor. De manera general, en cada una de las preguntas se obtuvieron resultados favorables, pero se debería indagar un poco más con otras preguntas acerca de las preocupaciones de las respuestas no favorables.

Aunque la puntuación fue generalmente alta no es del todo perfecta, una de las causas puede ser que las personas encuestadas esperan funcionalidades adicionales en la aplicación.

Un punto importante que no se abordó en la encuesta fue la formulación de una pregunta relacionada con el género de la persona, se hubiera podido saber más de la percepción teniendo en cuenta el género.

No se encontró la reglamentación acerca de calcomanías y adhesivos en el servicio público, según si la hay, podría ser una limitación del uso de la aplicación.

Haber elegido una aplicación web en vez de una aplicación móvil fue provechoso, pues el QR puede ser leído desde cualquier celular inteligente y la información del taxi puede ser mostrada en cualquier navegador sin necesidad de descargar ninguna aplicación.

El stack MERN es muy usado y cuando se presentan problemas en la programación se pueden solucionar de manera relativamente fácil porque en la web se encuentra mucha información, su comunidad es muy activa. Cuando se usan las tecnologías del MERN prácticamente se garantiza que las aplicaciones se vuelvan muy flexibles y escalables. Además, son de código abierto y ayuda a un proceso de diseño y programación modularizado lo que permite dividir la aplicación en módulos o componentes haciendo que se pueda reusar código, que este sea más legible y que

se dé fácil escalabilidad, pues así se puede agregar o quitar funciones sin afectar toda la aplicación.

Para ahorrar recursos económicos el despliegue de la aplicación y su base de datos se hicieron en un modo con características que son limitadas, pero que son más que suficientes para hacer pruebas de funcionalidad de la aplicación.

Usar Semantic UI es muy ventajoso ya que sus componentes son sensibles y por esto se adaptan a diferentes tamaños de pantalla.

El uso de todas las tecnologías mencionadas en la realización de este informe permite que la aplicación tenga un buen rendimiento, haciendo que al momento de hacer cualquier solicitud a cualquiera de las páginas se vea un cargue fluido, pero sobre todo rápido.

BIBLIOGRAFÍA

Grandas Enrique, Bautista Andrés (2021). Confianza en el uso de servicio de taxis públicos y su vínculo con la lealtad de marca.
https://repository.cesa.edu.co/handle/10726/4177

Cala Juliana (2019). Factores que influyen en la selección modal del taxi y servicios de ridesourcing en Bogotá.
https://repositorio.uniandes.edu.co/handle/1992/35036

Migliorata Emiliano, Dahl Juan Ricardo (2019). Diseño e Implementación de una aplicación web orientada a carpooling aplicando prácticas y tecnologías innovadoras maximizando la eficiencia y calidad del producto.
https://www.ridaa.unicen.edu.ar/items/184cd304-03fc-407a-afe0-0c7181379e89

Hernández Rubén (2020). Plataforma Web de recolección de datos del grupo de investigación de ciencias básicas del sistema músculo-esquelético de la Universidad El Bosque.
https://repositorio.unbosque.edu.co/handle/20.500.12495/9172

Lemus, N. C., & Gómez García, I. (2016). Incorporación de los códigos QR en la Educación Física en Secundaria Incorporating QR codes in Physical Education in Secondary (Vol. 29).

“MERN Stack Explained” [En Linea] Disponible: https://www.mongodb.com/mern-stack

“Getting Started Railway” [En Linea] Disponible: https://docs.railway.app/getting-started

“A Step-by-Step Guide: Deploying on Netlify” Disponible: https://www.netlify.com/blog/2016/09/29/a-step-by-step-guide-deploying-on-netlify/

ANEXO

Los códigos completos están en el repositorio de GitHub en los siguientes links.

Servidor: https://github.com/SebG15/servergr

Cliente: https://github.com/SebG15/client-gr