Desarrollo aplicación web responsive que gestiona la reserva de espacios de trabajo y da métricas de asistencia en las nuevas instalaciones de Cidenet S.A.S

Alejandro Estrada Moscoso

Informe de trabajo de grado como requisito para optar al título de:
Ingeniería de Telecomunicaciones

Tutor
Natalia Gaviria Gómez
Profesora Facultad de Ingeniería

Universidad de Antioquia
Facultad de Ingeniería
Pregrado
Medellín, Antioquia, Colombia
2022

Cita	(Estrada Moscoso, 2022)
Referencia **Estilo APA 7 (2020)**	Estrada Moscoso, A., (2022). *Desarrollo aplicación web responsive que gestiona la reserva de espacios de trabajo y da métricas de asistencia en las nuevas instalaciones de Cidenet S.A.S*, Semestre de Industria. Universidad de Antioquia, Medellín, Antioquia.

Índice de Contenido

1. Resumen

Este proyecto nace de la iniciativa de tener un sistema de control de reservas de espacios de trabajo autogestionable que mejore la experiencia de usuario hacia los colaboradores de Cidenet S.A.S. Debido a la inauguración de las nuevas instalaciones en la ciudad de Medellín y al constante crecimiento del personal en la empresa**,** se propuso un sistema de tickets secuencial que

el personal del área de infraestructura atiende diariamente. Esta situación a lo largo de los meses, se volvió tediosa, hasta llegar al punto de generar malestar entre los colaboradores por los retrasos en las respuestas. Esto también limitaba la capacidad de acción del área de infraestructura en otras actividades que fueran de mayor valor para la empresa.

En adición, los impactos de la pandemia de la COVID19, fortalecieron los esquemas de bioseguridad en todas las empresas que tienen sedes presenciales, Cidenet S.A.S no se quedó atrás, por lo que se solicitó tener en el proyecto, métricas de asistencia poblacional durante los días laborales, esto con el fin de tener una trazabilidad de las personas y en caso de contagio, tener los registros con exactitud e informarle a los asistentes las medidas de bioseguridad a seguir.

Durante el desarrollo de la práctica, se implementó una plataforma web responsive bajo la arquitectura de microservicios, que logró resultados positivos en los tiempos de solicitud. Previamente, la mayoría de la población experimentaba tiempos de respuesta de las reservas de mínimo 5 minutos hasta más de una hora. Con este desarrollo, se mejoraron todos estos índices, reduciendo los periodos de confirmación a menos de un minuto. El registro de asistencia dió un gran valor informativo sobre el flujo del personal en las instalaciones y se percibió una gran aceptación por parte de los directores de las diferentes áreas de la empresa sobre el trabajo realizado.

Palabras clave: Web, Responsive, Arquitectura microservicios, Infraestructura, Gestión de reservas

2. Introducción

Cidenet S.A.S es una empresa fundada en 2012, que se dedica a prestar servicios de desarrollo de software a la medida bajo un enfoque de desarrollo ágil. Actualmente, brinda sus servicios comerciales en diferentes locaciones geográficas: Sudamérica, Centroamérica y Norteamérica. La empresa cuenta con una amplia experiencia en tecnologías frontend y backend que garantizan un desarrollo de alta calidad en los servicios brindados a los clientes. Utilizando el marco ágil SCRUM, Cidenet busca entregar a los clientes productos de software de valor de forma iterativa e incremental (Sprints de máximo tres semanas), que estén alineados con los objetivos estratégicos del negocio y con una entrega a tiempo de los productos que evidencien el compromiso de la empresa hacia sus clientes.

Desde inicios de la pandemia de la COVID19, la mayoría de empresas a nivel nacional cesaron sus actividades presenciales hacia una nueva tendencia orientada al teletrabajo. Cidenet S.A.S, como empresa de software, contaba con instalaciones para que los desarrolladores, gerentes, coordinadores y demás personal asistiera a este lugar y poder realizar sus actividades diarias. Con

los emergentes cambios en la modalidad de trabajo, Cidenet S.A.S se adaptó fácilmente hacia un entorno virtualizado que ayudó a la mayoría de personas en la empresa a realizar sus deberes tal cual lo hacían de manera presencial. Estos grandes cambios facilitaron la apertura de más vacantes de trabajo tanto a nivel nacional como internacional. De esta forma, la empresa pasó de contar con 50 empleados a tener más de 100 personas vinculadas.

A comienzos de 2022, Cidenet S.A.S contaba con más del doble de personal del que había a principios de 2019. Debido a esta situación y a la falta de espacios de trabajo en la antigua sede, la empresa decidió adquirir nuevas instalaciones para ofrecer mayor comodidad y un ambiente laboral innovador y ameno para los trabajadores. Una de las problemáticas que se preveían debido al incremento constante del personal, era poder contar con las estaciones de trabajo suficientes para los días en que los trabajadores decidieran movilizarse a la sede. En primera instancia, se tomó la medida de que mediante unos tickets generados por los trabajadores en la plataforma de Cidenet y a través de la mesa de trabajo de la intranet, un trabajador de gestión humana se encargaría de reservar estos espacios de trabajo. Al ver que esta era una solución provisional y poco escalable, se propuso generar una plataforma web responsive para controlar todo el flujo de reservas de espacios de trabajos en todo el entorno físico de Cidenet S.A.S, que genere notificaciones de confirmación vía email y sea capaz de generar y/o quitar espacios de trabajo según las necesidades internas de la empresa. Adicionalmente, la solución debe contar con un componente de historial de asistencia que dictamine las personas que asistieron a la sede en algún día en específico, con el fin de tener un rastreo cuantitativo de las personas y así establecer un cerco epidemiológico en caso de que se presentara algún caso de contagio o sospecha de COVID-19.

Por lo tanto, en este semestre de industria se realizó una aplicación web para satisfacer las necesidades actuales de Cidenet S.A.S, con respecto a los espacios de trabajo en sus nuevas instalaciones. Dicha implementación se enmarca en los parámetros de calidad que exige la empresa en la entrega de software de calidad, usando metodología ágil SCRUM.

3. Objetivos

3.1 Objetivo General

Desarrollar una aplicación web responsive mediante una arquitectura de microservicios, uso software open source, servicios en la nube y bajo el modelo de trabajo ágil SCRUM, donde Cidenet S.A.S tendrá de una manera escalable el control de los espacios de trabajo en sus nuevas instalaciones.

3.2 Objetivos Específicos

- Realizar el levantamiento de requisitos mínimos de la aplicación, donde se establezcan claramente los condicionantes, alcances e impacto dentro de la empresa y sus futuros usuarios.
- Establecer historias de usuario bajo la metodología de SCRUM para tener una trazabilidad en el desarrollo de software.
- Aplicar las múltiples utilidades que brinda el framework React en la estructuración del frontend de la aplicación web y la implementación del backend con java spring boot.
- Seleccionar el proveedor de servicios en la nube que mejor costo-beneficio brinde en procesos de hosting, base de datos, cloud functions, servicios de mensajería y autenticación.
- Evaluar el desempeño funcional de la aplicación con los líderes técnicos e integrantes del grupo de trabajo.

4. Marco Teórico

Para la realización del presente proyecto, se utilizaron diferentes ambientes de desarrollo frontend y backend que facilitaron su implementación y despliegue, así como diferentes técnicas organizacionales para obtener un software de calidad, cumpliendo con los requisitos exigidos y acertando los tiempos establecidos.

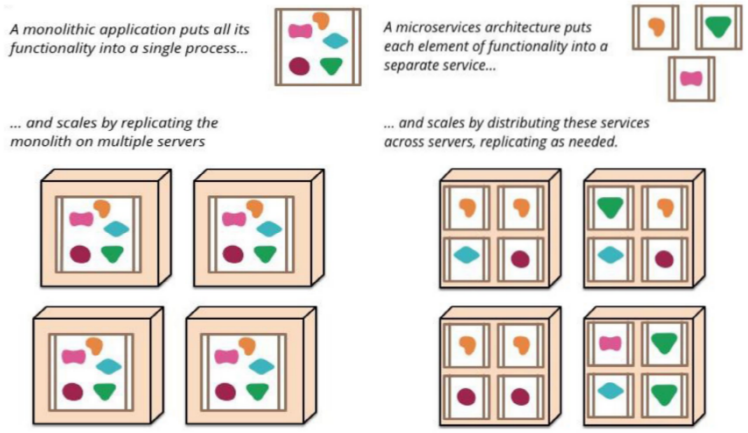
El proyecto fue estructurado mediante una arquitectura de microservicios, que separa los servicios de la aplicación por parte del cliente así como los recursos de consumo, en entidades que se pueden granular tanto como sea necesario. En la parte del frontend se usó el framework open source de React Js, especializado en estructurar la parte visual de la aplicación mediante componentes; para el backend se implementó un microservicio desarrollado en Java spring-boot; los servicios en la nube mayoritariamente fueron prestados por AWS y Google. Finalmente, para los procesos de desarrollo ágil se utilizó SCRUM, puesto que es la metodología definida por Cidenet S.A.S para el desarrollo de software de calidad.

4.1 Arquitectura del proyecto

Arquitectura de microservicios

Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose entre sí mediante protocolos de comunicación sincrónica como HTTP/HTTPS.[1]

Figura 1. Comparación Abstracta entre arquitectura tradicional ó monolítica vs Arquitectura de microservicios



Fuente: González, I. (2018). Market Cart App: aplicación móvil para la gestión de compra de víveres en línea. [Consultado: 21 de Agosto de 2022]. Disponible en: https://revistas.udistrital.edu.co/index.php/tia/article/view/9687/pdf

Una arquitectura de microservicios promueve el desarrollo de aplicaciones compuestas por unidades independientes, autónomas, modulares y auto-contenidas, lo cual difiere de la forma tradicional o monolítica, como se evidencia en la figura 1 (en la parte izquierda se muestra la aplicación replicada en servidores como un "todo", diferente a la granularidad individual de servicios que se presencia en la parte derecha). Una de las ventajas de utilizar microservicios es la capacidad de gestionar una aplicación grande como un conjunto de pequeñas aplicaciones (microservicios) que se pueden desplegar, escalar, manejar y visualizar de forma independiente [1].

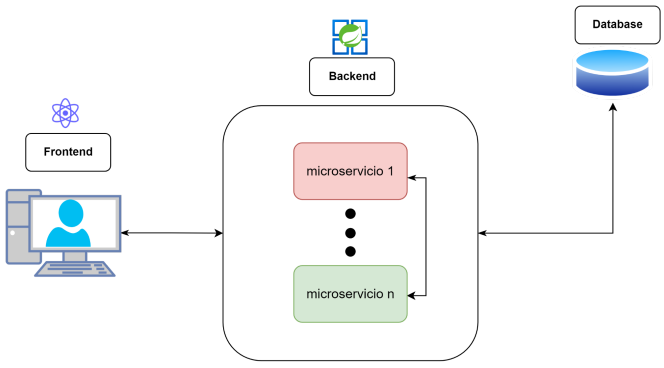


Figura 2. Estructura General Arquitectura de microservicios

Otra característica de esta arquitectura es que permite desarrollar la aplicación en distintos lenguajes de programación, puesto que los servicios van a estar aislados y se podrán comunicar con mecanismos comunes. En la figura 2, se muestra una arquitectura de microservicios compuesta por React y Java spring boot para frontend y backend respectivamente, adicionalmente esto no limita su desarrollo en múltiples lenguajes como Angular y node o Vue y

Python. Esto hace que los servicios sean más especializados, es decir, que al poder elegir distintos tipos de lenguaje, se podrán resolver problemas más específicos de una forma más eficiente.[2]

Los microservicios permiten a las empresas gestionar las aplicaciones de código base grande usando una metodología más práctica donde las mejoras incrementales son ejecutadas por pequeños equipos en bases de código y despliegues independientes [1].

4.2 Tecnologías de Desarrollo

4.2.1 React Js

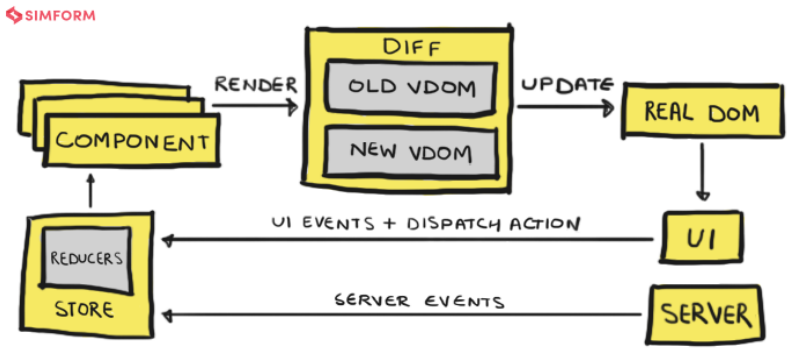
React Js es una librería JavaScript de frontend que permite construir interfaces de usuario. Se caracteriza por simplificar el uso convencional de Vanilla Js, alto rendimiento en renderizado y adaptable al aumento de componentes en la aplicación. Fue desarrollada por Facebook y al igual que otros frameworks de frontend como Angular, está diseñada para la creación de aplicaciones web de una sola página. Al ser un framework declarativo, diseña vistas simples para cada estado de la aplicación, y React se encarga de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. Las vistas declarativas hacen que el código sea más predecible, por lo tanto, fácil de depurar [3][4].

```jsx
import React from 'react'

export const ExampleComponent = () => {


    const [searchBarText, handleInputChange] = useForm({
        search: '',
    });

    const { search } = searchBarText;

    const onSubmit = (e) => {
        e.preventDefault(e);
    }

  return (
        <form>
            <input
                type="text"
                placeholder="Busqueda por filtro"
                className="form-control"
                id="search"
                name="search"
                value={search}
                onChange={handleInputChange}
            />
        </form>
    );
}
```

Figura 3. Ejemplo Componente Funcional en React

En la figura 3, se expone un ejemplo básico de un componente funcional de React Js que puede ser reutilizado en múltiples lugares de la aplicación. Este se estructura por: librerías, definición del componente, lógica interna y resultado en formato JSX del componente. Dadas las bondades que posee React Js, y gracias al rendimiento que brinda en las aplicaciones desarrolladas, ha sido considerada también para ser utilizada en aplicaciones móviles sin cambiar su funcionalidad [3].

Figura 4. Estructura base de React



Fuente: Kasundra, P. (2020). React Architecture Best Practices and Tips from Community Experts. [Consultado: 21 de Agosto de 2022]. Disponible en: https://www.simform.com/blog/react-architecture-best-practices.

En la figura 4, se muestra la estructura base de React Js, que consta de los siguientes componentes: flujos de datos, DOM (Document Object Model) virtual y representaciones condicionales. Dichas características hacen que las funcionalidades y librerías de React mejoren en términos de rendimiento y estructuración limpia del código en comparación con el antiguo paradigma de programación en javascript, Vanilla Js [3].

4.2.2 Java Spring Boot

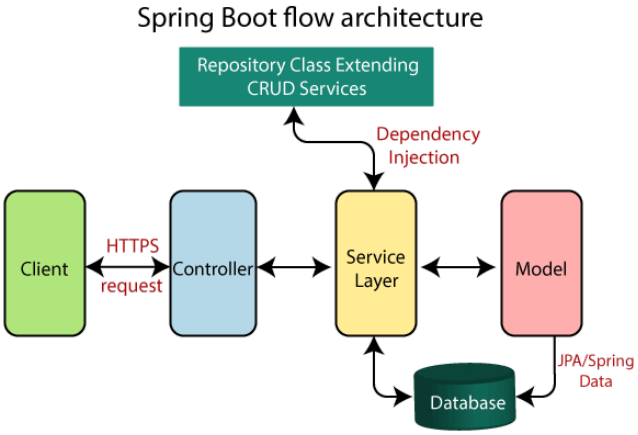
Spring es un marco de trabajo para aplicación web, que se ha popularizado entre los programadores. Con el tiempo, los proyectos basados en él fueron generando inconvenientes por un gran número de archivos XML que tenían que estar correctamente configurados para que los componentes individuales y la aplicación funcionaran correctamente [6][5].

Esto ha causado muchas frustraciones entre los programadores al invertir tiempo que puede ser valioso en la lógica de negocios. Por lo tanto, se creó Spring Boot, un módulo desarrollado a partir de Spring, que nace con la finalidad de simplificar aún más la programación de aplicaciones basadas en el framework [6].

Spring Boot simplifica significativamente tareas convencionales como el despliegue del servidor, el cual ya se encuentra embebido dentro del proyecto, que por defecto es un Tomcat permitiendo

a este que se ejecute en un Jar. Las configuraciones del ambiente de desarrollo se configuran automáticamente. De igual manera, Spring Boot permite usar las herramientas de gestor de proyectos como Maven y Gradle mediante la administración de dependencias [6].

Figura 5. Arquitectura base de Java Spring Boot



Fuente: Javatpoint (2022). Spring Boot Architecture. [Consultado: 21 de Agosto de 2022]. Disponible en: https://www.javatpoint.com/spring-boot-architecture.

En la figura 5, se muestra la estructura general de Java spring boot en componentes. La capa de servicio y modelo maneja los procesos relacionados con la captura e inserción de información en la base de datos, la correcta estructura de los datos y cómo estos se implementan internamente. En la sección del controlador, están definidos los puntos de acceso desde el cliente hacia los servicios y lógica interna del backend.

4.3 Proveedores de servicios en la nube

4.3.1 AWS

Amazon Web Services ofrece un amplio conjunto de productos globales basados en la nube, incluidos recursos informáticos, almacenamiento, bases de datos, análisis, redes, dispositivos móviles, herramientas para desarrolladores, herramientas de administración, IoT, seguridad y aplicaciones para empresas: en diferido, disponibles en segundos, con precios de pago por uso. AWS dispone de más de 200 servicios, desde almacén de datos a herramientas de implementación, y desde directorios hasta entrega de contenido. Los nuevos servicios se aprovisionan rápidamente, sin los gastos de capital iniciales, para que grandes corporaciones, startups, pequeñas y medianas empresas y clientes del sector público tengan acceso a los recursos básicos que necesitan para responder con rapidez a los cambiantes requisitos empresariales. [7]

4.3.2 Google - OAuth2

El marco de autorización de OAuth 2.0 es un framework que permite que un tercero obtenga acceso limitado a un servicio HTTP, ya sea en nombre de un propietario de recursos, organizando una interacción de aprobación entre el propietario del recurso y el servicio HTTP, o permitiendo que el aplicación de terceros obtenga acceso en su propio nombre. Este reemplaza y deja obsoleto el protocolo OAuth 1.0.[8]

4.4 Sistema de Versionamiento

4.4.1 Git

Git es un un sistema distribuido de control de versiones, gratuito y de código abierto bajo licencia GPLv2. Fue diseñado originalmente por Linus Torvalds, el creador de Linux. Git fue concebido con la idea de ofrecer un gran rendimiento, ocupar menos espacio en disco y evitar la necesidad de un servidor central para que pudiera ser distribuido. De hecho, esto último es lo que hacía que fuese realmente novedoso respecto a otras alternativas del momento como *Subversion*, que tenían una solución más centralizada.[9]

Con todas esas ventajas, Git se convirtió en un sistema de confianza a la hora de mantener versiones de aplicaciones con una gran base de código. De esta forma, en 2005 se empezó a desarrollar para ser utilizado con el Kernel de Linux. De ahí hasta el día de hoy, Git ha seguido evolucionando y se ha establecido como el sistema de control de versiones más usado en el mundo de la programación y el desarrollo de software.[9]

4.4.2 GitLab

GitLab es un software de control de versiones basado en git, que agrega funciones adicionales como el seguimiento de errores de administración de sucursales y la integración continua. Está desarrollado en rubí. Un repositorio Git en la nube solía ser el principal punto de venta de GitLab. Pero la plataforma ha evolucionado más allá de sus simples orígenes.[10]

Hoy en día, GitLab ofrece una amplia gama de características DevOps, como la integración continua, la seguridad e incluso herramientas de despliegue de aplicaciones.También ofrece herramientas esenciales de gestión de proyectos para supervisar y controlar a los miembros del equipo. [11]

4.5 Base de Datos

4.5.1 Base de datos relacional

Una base de datos relacional es una base de datos que cumple con el modelo relacional, que permite establecer interconexiones entre datos, y a través de dichas conexiones relacionar registros de las tablas que componen la base de datos. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, en los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en la persistencia de información.[12]

A las relaciones, en diferentes lenguajes de programación para conjuntos de datos, se les denomina tablas. Cada tabla tiene una serie de columnas (son los atributos); cada columna tiene un nombre distinto y es de un tipo de datos (entero, real, carácter, fecha, etc.). En las tablas se insertan filas (son las tuplas), que después se pueden consultar, modificar o borrar.[13]

Características[12]*:*

- Una base de datos relacional se compone de varias tablas o relaciones.
- No pueden existir dos tablas con el mismo nombre ni registro.
- Cada tabla es a su vez un conjunto de registros (filas y columnas).
- La relación entre una tabla padre y un hijo se lleva a cabo por medio de las claves primarias y ajenas (o foráneas).
- Las claves primarias son la clave principal de un registro dentro de una tabla y éstas deben cumplir con la integridad de los datos.
- Las claves ajenas se colocan en la tabla hija, contienen el mismo valor que la clave primaria del registro padre; por medio de éstas se hacen las relaciones.

4.5.2 SQL

SQL (Structured Query Language) es el lenguaje estándar de las bases de datos relacionales. Es un lenguaje declarativo que permite especificar diversos tipos de operaciones, a su vez es capaz de conjugar las operaciones del álgebra y el cálculo relacional con operadores adicionales, y definir así las consultas para recuperar o modificar información de bases de datos, así como hacer cambios en ellas.
SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. En conjunto, se dispone de instrucciones para definir (crear y modificar el esquema), mantener (insertar, actualizar,eliminar) y finalmente, consultar registros en las tablas de modelo relacional.[14]

4.5.3 PostgreSQL

PostgreSQL es un servidor de base de datos objeto relacional libre, ya que incluye características de la orientación a objetos, como son la herencia, los tipos de datos, las funciones, las restricciones, los disparadores, las reglas y la integridad transaccional, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y

organizaciones comerciales que trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).[15]

4.6 Conceptos técnicos de desarrollo

4.6.1 Diseño web Responsive

El diseño web responsive ó también conocido como adaptativo, se describe como una técnica de diseño y desarrollo web que, mediante el uso de estructuras e imágenes fluidas, así como de media-queries en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario. Toma las mejores prácticas para aplicarlas en la construcción de sitios, logrando buena calidad en las aplicaciones. La idea es que un solo sitio sea no sólo adaptable a las características del recurso, sino que llegue a ser adaptativo.[16]

Figura 6. Ejemplo de diseño web responsive

Fuente: R. Penciarolli (2018). Diseño web responsive. [Consultado: 23 de Agosto de 2022]. Disponible en: https://imgbiblio.vaneduc.edu.ar/fulltext/files/TC130197.pdf

El diseño responsive mejora la experiencia del usuario al permitir que accedan al sitio web desde cualquier dispositivo, ya sea desde tablets, móviles, notebooks, o pc de escritorio, lo cual significa que los usuarios tendrán una grata experiencia al visitar el sitio. Cabe destacar que los sitios web responsive, generalmente, cargan más rápido, lo cual aumenta su posicionamiento en los buscadores web. [17]

El diseño de la información es variable y relativo, denominado fluido. La composición se presenta en forma dinámica, puesto que cada elemento se ajusta a las condiciones técnicas de despliegue. Para hacer un diseño web adaptativo se debe cumplir con los siguientes aspectos:[16]

1. Diseño fluido con cuadrículas flexibles o fluid grids.
2. Media Queries.
3. Imágenes, objetos, videos o medios similares flexibles.

4. Fuentes tipográficas con valores relativos.

4.6.2 REST

El término REST proviene de la tesis doctoral de Roy Fielding, publicada en el año 2000, y significa Representational State Transfer. REST es un conjunto de restricciones que, cuando son aplicadas al diseño de un sistema[18], tratan de minimizar la latencia y las comunicaciones de la red y, al mismo tiempo, maximizar la independencia y escalabilidad de las implementaciones de los componentes[19]. Las seis limitaciones de la REST incluyen lo siguiente:

1. Cliente-Servidor: Requiere que un servicio ofrezca una o más operaciones, y que los servicios esperen que los clientes soliciten dichas operaciones.[19]
2. Sin estado: Tiene que ser sin estado, es decir, no hay necesidad de que los servicios guarden las sesiones de los usuarios (cada petición al servicio tiene que ser independiente de las demás).[18]
3. Caché: Requiere que las respuestas se etiqueten claramente como con caché o son caché.[19]
4. Interfaz Uniforme: Requiere que todos los proveedores del servicio y consumidores dentro de una arquitectura que cumple con la REST compartan una sola interfaz común para todas las operaciones.[19]
5. Sistema en capas: Debe soportar la escalabilidad [18].

4.6.3 JSON

JSON (Javascript Object Notation, por sus siglas en inglés) es un formato ligero para intercambio de datos que surge como alternativa a XML en AJAX. Se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia, cuando la fuente de datos es explícitamente de confianza y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente.[20] Es un formato basado en un subconjunto de la sintaxis de JavaScript: literales de matrices y objetos. Como usa la sintaxis JavaScript, las definiciones JSON pueden incluirse dentro de archivos JavaScript y acceder a ellas sin ningún análisis adicional como los necesarios con lenguajes basados en XML.[21]

JSON está compuesto de dos estructuras[22]:

1. Una colección de pares nombre/valor
2. Una lista organizada de valores

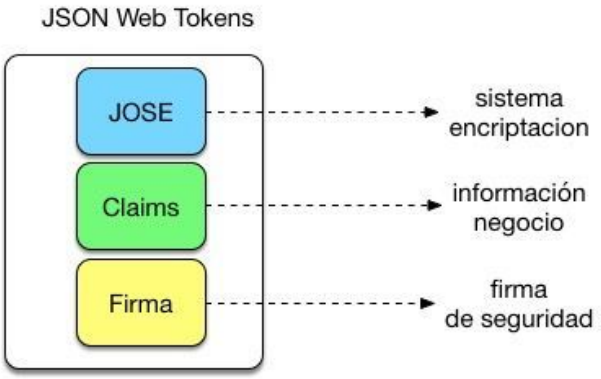
```
{
    "startDate": "2022-08-05 08:00",
    "endDate": "2022-08-05 17:00",
    "user": "11",
    "workStation": "19"
}
```


Figura 7. Formato básico de una estructura tipo JSON

4.6.4 JSON Web TOKEN

Un JSON Web Token es un token de acceso estandarizado que permite el intercambio seguro de datos entre dos partes. Contiene toda la información importante sobre una entidad, lo que implica que no hace falta consultar una base de datos ni que la sesión tenga que guardarse en el servidor (sesión sin estado). Por este motivo, los JWT son especialmente populares en los procesos de autenticación. Con este estándar es posible cifrar mensajes cortos, dotarlos de información sobre el remitente y demostrar si este cuenta con los derechos de acceso requeridos.[23]

Figura 8. Estructura de JWT



Fuente: C. Álvarez (2017). Introducción a JSON Web Token y la seguridad. [Consultado: 23 de Agosto de 2022]. Disponible en: https://www.arquitecturajava.com/introduccion-a-json-web-token/

En la figura anterior, se describe la estructura de un JSON Web TOKEN[24]:

1. En la parte superior se encuentra el JOSE o Javascript Object Signing and Encryption y define cuál es la tecnología criptográfica que se va a aplicar al token para proteger la información.
2. La parte media se denomina JWT Payload o JWT Claims y almacena la información de negocio que encripta el token.
3. La última parte es la firma JWT que se encarga de dar validez al token.

4.6.4 SMTP

El Simple Mail Transfer Protocol (SMTP) o "protocolo para transferencia simple de correo", es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA, teléfonos móviles, etc).[25] SMTP es un protocolo independiente del subsistema de transmisión usado. Necesita que el subsistema de transmisión ponga a su disposición un canal de transmisión fiable y con entrega ordenada, con lo cual el uso del protocolo TCP en la capa de transporte es lo adecuado. Para que dos sistemas intercambien correo mediante el protocolo SMTP, no es necesario que exista una conexión interactiva, ya que este protocolo usa métodos de almacenamiento y reenvío de mensajes.[26]

4.7 Metodología de Desarrollo

Scrum

Scrum es un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos. Scrum se basa en el empirismo y el pensamiento Lean. El empirismo afirma que el conocimiento proviene de la experiencia y de la toma de decisiones con base en lo observado. El pensamiento Lean reduce el desperdicio y se enfoca en lo esencial [27].

Scrum emplea un enfoque iterativo e incremental para optimizar la previsibilidad y controlar el riesgo. A su vez, involucra a grupos de personas que colectivamente tienen todas las habilidades y experiencia para hacer el trabajo y compartir o adquirir dichas habilidades según sea necesario. Dicho grupo se compone de [28]:

Scrum Master: Responsable de gestionar el proceso Scrum y ayudar a eliminar impedimentos que puedan afectar la entrega del producto [28].
Product Owner: Responsable de maximizar el valor del producto resultante del trabajo del Scrum Team [27].
Developers: Responsables de desarrollar el producto, auto-organizándose y auto-gestionándose para conseguir entregar un incremento de software al final del ciclo de desarrollo [28].

4.8 Modalidad de Trabajo

Teletrabajo

Según la «Guía de Teletrabajo» del gobierno federal de Estados Unidos, califican como teletrabajo aquellos arreglos de trabajo en los cuales el empleado atiende regularmente algunas tareas desde su casa o en lugares de trabajo acondicionados cerca de su casa. En Colombia, el teletrabajo se considera una forma de organización laboral, que consiste en el desempeño de actividades remuneradas o la prestación de servicios a terceros utilizando como soporte las tecnologías de información y comunicación (TIC) para el contacto entre el trabajador y la empresa, sin requerir la presencia física del trabajador en un sitio específico de trabajo.[29]

4.9 Concepto de negocio

MVP

Un MVP (Mínimo Producto Viable) es un concepto lanzado por Eric Ries en 2011, el cual se define como la versión de un nuevo producto con la menor cantidad de funciones posibles, que permite al equipo de desarrollo recolectar información para validar respuestas del cliente, utilizando la menor cantidad de recursos posible (tiempo,dinero y esfuerzo). Suele confundirse al MVP con un prototipo. Es importante detallar que un MVP no es un prototipo, sino un producto que cuenta con funcionalidades mínimas para ser lanzado al mercado. [30]

5. Metodología

5.1 Etapa 1: Establecimiento de historias de usuario bajo la metodología de SCRUM

Actividad 1.1
Se definieron las historias de usuarios tanto del frontend como del backend, con el fin de tener una estimación de tiempo para la realización del MVP.

Tabla 1. Historias de usuario del frontend

Cantidad	HU	Módulo	Descripción
1	40	Frontend	Creación del proyecto con "create-react-app"
2	41	Frontend	Maquetación del frontend
3	42	Frontend	Adición de dependencias de estilo como bootstrap
4	43	Frontend	Adición de validadores de código e identación
5	44	Frontend	Enrutamiento de páginas
6	45	Frontend	Configuración de rutas privadas y públicas
7	46	Frontend	Creación del componente de login
8	47	Frontend	Prueba de concepto del login con google desde el front
9	48	Frontend	Prueba de concepto de reserva de espacios de trabajo

10	49	Frontend	Definición del componente home
11	50	Frontend	Adición de material UI
12	51	Frontend	Desarrollo estructura de home y formulario de reserva de espacios de trabajo
13	52	Frontend	Desarrollo de flujo creación de reserva con persistencia de datos locales
14	53	Frontend	Creación del componente de reservas
15	54	Frontend	Ajustes de diseño responsive para el componente de reservas
16	55	Frontend	Creación del diseño de las cards de las reservas
17	56	Frontend	Implementación de modales de confirmación y cancelación de reservas
18	57	Frontend	Creación del componente de estadísticas de asistencia
19	58	Frontend	Prueba de concepto de la librería de chart js para las gráficas
20	59	Frontend	Generación de la modal que filtra las consultas
21	60	Frontend	Prueba de concepto con librería de creación de documentos de excel
22	61	Frontend	Implementación de la librería de creación de documentos de excel con la modal de filtro de consultas
23	62	Frontend	Conexión con los endpoints de manera local
24	63	Frontend	Pruebas funcionales de la aplicación
25	64	Frontend	Conexión del front con las rutas públicas del back
26	65	Frontend	Creación del bucket s3 para almacenar los archivos del front y tener una ruta pública de acceso

Tabla 2. Historias de usuario del backend

Cantidad	HU	Módulo	Descripción
27	66	Backend	Uso de maquetas base de java spring boot
28	67	Backend	Maquetación del backend
29	68	Backend	Adición de dependencias
30	69	Backend	Definición del controlador de reservas
31	70	Backend	Pruebas de peticiones locales hacia el controlador de reservas
32	71	Backend	Se definen todos los puntos de acceso del controlador de reservas
33	72	Backend	Se definen los servicios y funcionamiento lógico de las reservas
34	73	Backend	Se configura el AWS Secret manager
35	74	Backend	Conexión del backend con la base de datos Amazon RDS
36	75	Backend	Definición de los modelos y DTO's para las peticiones a la base de

			datos
37	76	Backend	Pruebas de persistencia de información en la base de datos
38	77	Backend	Definición de las Queries SQL para filtrar la información de la base de datos
39	78	Backend	Definición del controlador de espacios de trabajo
40	79	Backend	Pruebas de peticiones locales hacia el controlador de espacios de trabajo
41	80	Backend	Se definen el punto de acceso del controlador de espacios de trabajo que dará las estaciones disponibles en las horas solicitadas
42	81	Backend	Definición de los modelos y DTO's para las peticiones a la base de datos para las estaciones de trabajo
43	82	Backend	Definición de las Queries SQL para filtrar la información de la base de datos para las estaciones de trabajo
44	83	Backend	Se define el punto de acceso del controlador de reservas para brindar la información sobre las estadísticas de asistencia y estado de las reservas
45	84	Backend	Definición de los modelos y DTO's para las peticiones a la base de datos para las estadísticas
46	85	Backend	Definición de las Queries SQL para filtrar la información de la base de datos para las estadísticas
47	86	Backend	Se adicionaron las dependencias de google OAuth2
48	87	Backend	Se implementó sistema de autenticación de google por lado del backend
49	88	Backend	Se desplegó en una instancia EC2 de Amazon el backend
50	89	Backend	Conexión entre el backend y frontend

Las dos anteriores tablas detallan las diferentes historias de usuario planteadas para llevar a cabo el MVP. Estas fueron susceptibles a cambios o adiciones a medida que se desarrolló el frontend y backend.

Actividad 1.2

Se analizó cada historia de usuario donde se verifica la dificultad de su implementación, componentes de los frameworks necesarios y alcance de dichas actividades.

Actividad 1.3

Se estimaron las historias de usuario, esto teniendo en cuenta los tiempos del sprint, que en el caso particular de Gestionappte (nombre interno del aplicativo) , fueron de dos semanas, aproximadamente unas 64 horas, descontando tiempos de reuniones con líderes, acompañamientos y asesorías con personal de Cidenet S.A.S.

Actividad 1.4

Se priorizaron las historias de usuario de mayor impacto en la aplicación, tanto las que iban ligadas al backend como al frontend.

Tabla 3. Priorización y definición de dificultad de HU's del frontend

Módulo	Descripción	Prioridad	Dificultad
Frontend	Creación del proyecto con "create-react-app"	ALTA	BAJA
Frontend	Maquetación del frontend	MEDIA	ALTA
Frontend	Adición de dependencias de estilo como bootstrap	MEDIA	BAJA
Frontend	Adición de validadores de código e identación	BAJA	MEDIA
Frontend	Enrutamiento de páginas	ALTA	MEDIA
Frontend	Configuración de rutas privadas y públicas	MEDIA	MEDIA
Frontend	Creación del componente de login	ALTA	ALTA
Frontend	Prueba de concepto del login con google desde el front	BAJA	ALTA
Frontend	Prueba de concepto de reserva de espacios de trabajo	BAJA	ALTA
Frontend	Definición del componente home	ALTA	ALTA
Frontend	Adición de material UI	MEDIA	BAJA
Frontend	Desarrollo estructura de home y formulario de reserva de espacios de trabajo	ALTA	ALTA
Frontend	Desarrollo de flujo creación de reserva con persistencia de datos locales	ALTA	MEDIA
Frontend	Creación del componente de reservas	ALTA	ALTA
Frontend	Ajustes de diseño responsive para el componente de reservas	MEDIA	MEDIA
Frontend	Creación del diseño de las cards de las reservas	BAJA	MEDIA
Frontend	Implementación de modales de confirmación y cancelación de reservas	BAJA	MEDIA
Frontend	Creación del componente de estadísticas de asistencia	ALTA	ALTA
Frontend	Prueba de concepto de la librería de chart js para las gráficas	MEDIA	BAJA
Frontend	Generación de la modal que filtra las consultas	ALTA	MEDIA
Frontend	Prueba de concepto con librería de creación de documentos de excel	MEDIA	MEDIA
Frontend	Implementación de la librería de creación de documentos de excel con la modal de filtro de consultas	ALTA	MEDIA
Frontend	Conexión con los endpoints de manera local	MEDIA	BAJA
Frontend	Pruebas funcionales de la aplicación	MEDIA	MEDIA

Frontend	Conexión del front con las rutas públicas del back	ALTA	MEDIA
Frontend	Creación del bucket s3 para almacenar los archivos del front y tener una ruta pública de acceso	ALTA	ALTA

Tabla 4. Priorización y definición de dificultad de HU's del backend

Módulo	Descripción	Prioridad	Dificultad
Backend	Uso de maquetas base de java spring boot	MEDIA	BAJA
Backend	Maquetación del backend	MEDIA	BAJA
Backend	Adición de dependencias	ALTA	MEDIA
Backend	Definición del controlador de reservas	ALTA	MEDIA
Backend	Pruebas de peticiones locales hacia el controlador de reservas	MEDIA	MEDIA
Backend	Se definen todos los puntos de acceso del controlador de reservas	MEDIA	ALTA
Backend	Se definen los servicios y funcionamiento lógico de las reservas	ALTA	ALTA
Backend	Se configura el AWS Secret manager	ALTA	MEDIA
Backend	Conexión del backend con la base de datos Amazon RDS	ALTA	MEDIA
Backend	Definición de los modelos y DTO's para las peticiones a la base de datos	MEDIA	BAJA
Backend	Pruebas de persistencia de información en la base de datos	MEDIA	MEDIA
Backend	Definición de las Queries SQL para filtrar la información de la base de datos	ALTA	ALTA
Backend	Definición del controlador de espacios de trabajo	ALTA	MEDIA
Backend	Pruebas de peticiones locales hacia el controlador de espacios de trabajo	MEDIA	MEDIA
Backend	Se define el punto de acceso del controlador de espacios de trabajo que dará las estaciones disponibles en las horas solicitadas	MEDIA	ALTA
Backend	Definición de los modelos y DTO's para las peticiones a la base de datos para las estaciones de trabajo	MEDIA	BAJA
Backend	Definición de las Queries SQL para filtrar la información de la base de datos para las estaciones de trabajo	ALTA	ALTA
Backend	Se define el punto de acceso del controlador de reservas para brindar la información sobre las estadísticas de asistencia y estado de las reservas	ALTA	ALTA

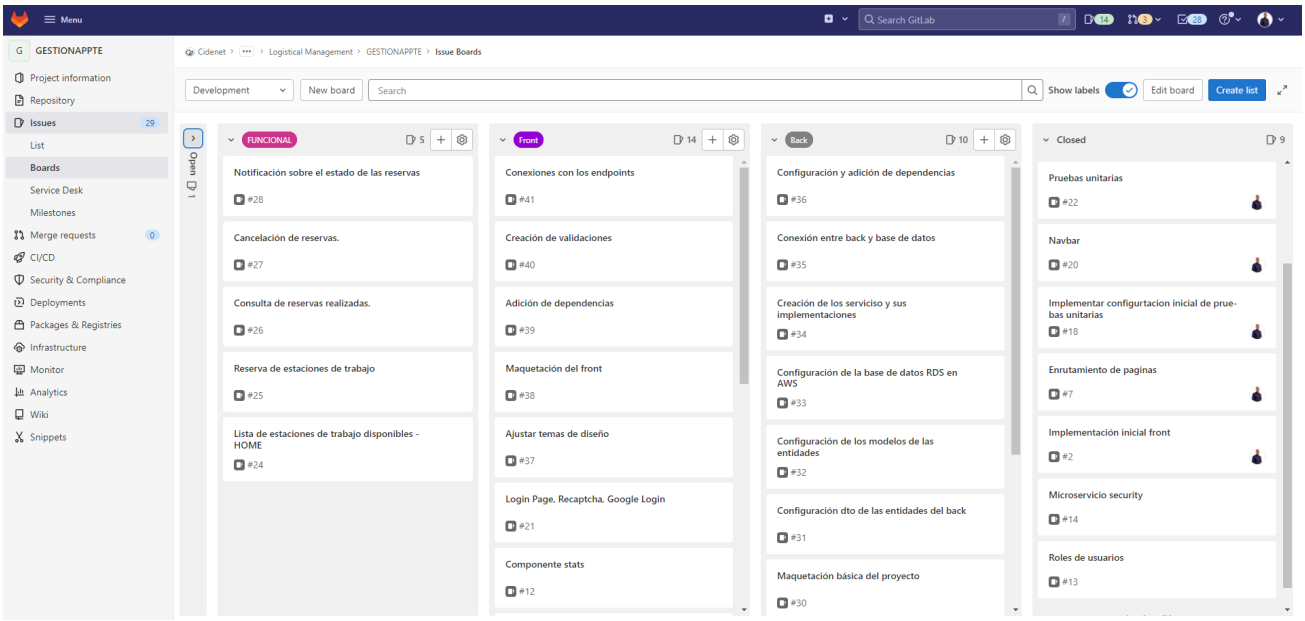
Backend	Definición de los modelos y DTO's para las peticiones a la base de datos para las estadísticas	MEDIA	BAJA
Backend	Definición de las Queries SQL para filtrar la información de la base de datos para las estadísticas	ALTA	ALTA
Backend	Se adicionaron las dependencias de google OAuth2	ALTA	MEDIA
Backend	Se implementó sistema de autenticación de google por lado del backend	ALTA	ALTA
Backend	Se desplegó en una instancia EC2 de Amazon el backend	MEDIA	MEDIA
Backend	Conexión entre el backend y frontend	MEDIA	BAJA

En las dos tablas anteriores, se definen la prioridad y dificultad que cada HU del módulo del frontend y backend. Estas estimaciones son susceptibles a cambios o adiciones a medida que se desarrolló el frontend y backend.

Actividad 1.5

Se generó un product backlog (listado de todas las tareas que se pretenden hacer durante el desarrollo de un proyecto) en Gitlab con el fin de tener registro de dichas historias de usuario.

Figura 9. Listado de Historias de Usuario (HUs) del product backlog

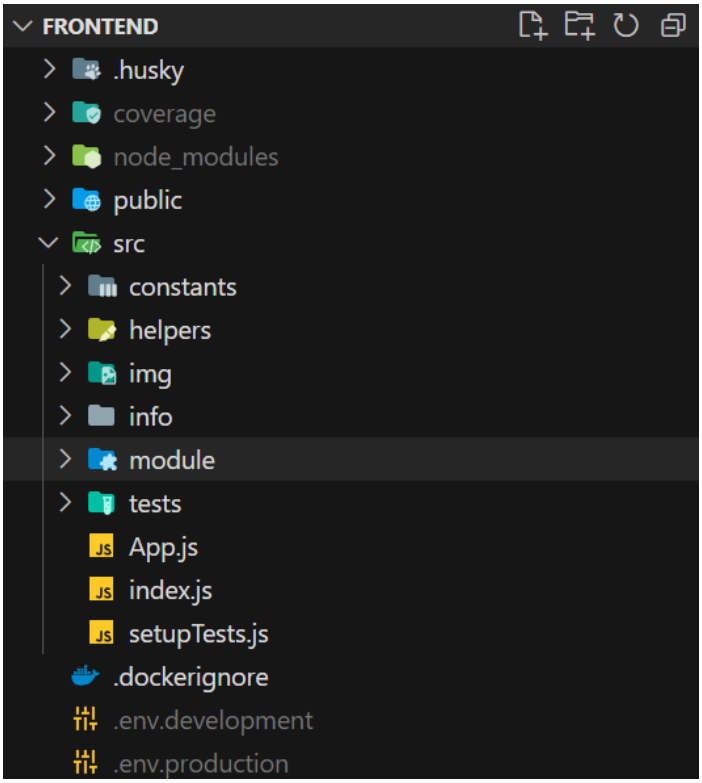


5.2 Etapa 2: Aplicación de las utilidades que brinda el framework para frontend React Js

Actividad 2.1
Se instaló el entorno básico de desarrollo de React Js, para esto fueron necesarios Node, NVM (node version manager, permite cambiar versión de node), NPX (genera un esquema básico de react).

Actividad 2.2
Se configuró un mapa de carpetas que sirviera para agrupar las diferentes funcionalidades de la aplicación.

Figura 10. Mapa de carpetas del frontend



En la imagen anterior, se evidencia el sistema de carpetas en el software Visual Studio Code que organiza modularmente las diferentes secciones del frontend.

Actividad 2.3
Se agregaron herramientas de indentación y organización estructural del código desarrollado. Para ello se utilizaron librerías como “prettier” y “lint-staged”.

Actividad 2.4
Se configuró el router de react para poder hacer el enrutado de rutas dentro de las diferentes partes de la aplicación.

Actividad 2.5
Se trabajó en conjunto con la diseñadora de Cidenet S.A.S con el fin de tener una línea gráfica que fuera fiel a los patrones característicos de la empresa.

Figura 11. Línea gráfica utilizada en Gestionappte

En la imagen anterior, se muestra la línea gráfica que se optó para el desarrollo de Gestionappte, esto con el fin de tener una clara relación con los colores distintivos de la empresa.

Actividad 2.6
Se empezaron a diseñar en alto nivel las diferentes interfaces y pantallas de la aplicación.

Actividad 2.7
Se desarrollaron las diferentes pantallas de la aplicación, tanto el formulario de registro a la plataforma como sus interfaces internas (formulario de reserva, reservaciones, estadísticas).

Actividad 2.8
Se probaron los diferentes módulos con datos de prueba con el fin de ver la funcionalidad y el alcance de las librerías y componentes funcionales de la aplicación.

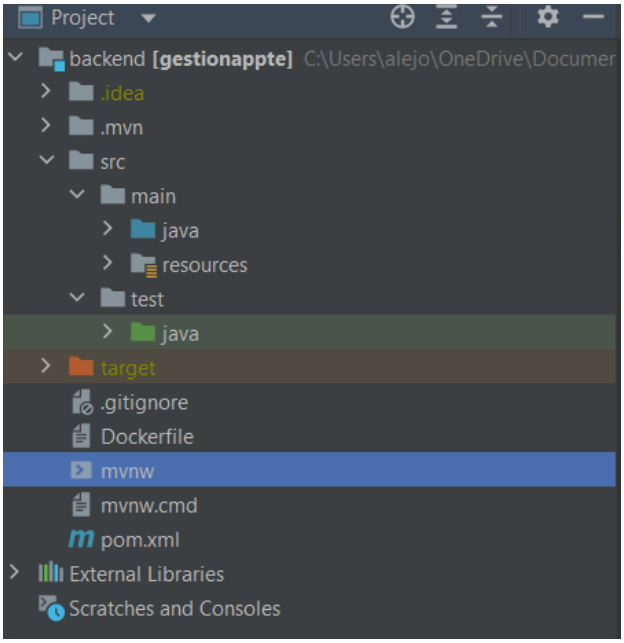
Actividad 2.9
Se registraron todos los cambios en diferentes commits que garantizarán de manera granular la persistencia del desarrollo del frontend.

5.3 Etapa 3: Aplicación de las utilidades que brinda Java Springboot para el backend

Actividad 3.1
Se instaló el entorno básico de desarrollo de Java Spring Boot, para esto fue necesario instalar el OpenJDK 15, Maven Wrapper y la dependencia de Spring Boot en su versión 2.3.3.

Actividad 3.2
Se mapeó el estándar de ordenamiento de carpetas, así como los diferentes niveles de la aplicación, esto con el fin de tener una estructura más organizada de las funcionalidades del proyecto.

Figura 12. Estructura de carpetas del backend



En la imagen anterior, se evidencia el sistema de carpetas en el software IntelliJ IDEA que organiza modularmente las diferentes secciones del backend.

Actividad 3.3
Se definió el tipo de base de datos a utilizar, en este caso, una base de datos relacional utilizando el motor SQL de PostgreSQL.

Actividad 3.4
Se configuró la base de datos en la nube mediante AWS, utilizando el conocido servicio de Amazon Relational Database Service, lo que permite tener base de datos en un entorno estable y administrado en términos de soporte por parte de AWS.

Actividad 3.5
Se desarrolló toda la parte de los controladores, es decir, se definieron en los endpoints que internamente, [“Crean las reservas”, “Listan las reservas”, “Confirman la reserva”, “Cancela la reserva”,”Consultan estadísticas”] y para las estaciones de trabajos junto con sus disponibilidades se definió otro controlador que se encargaba de comprobar y listar los puestos de trabajo libres.

Actividad 3.6
Se implementaron los DTO’s (Data Transfer Object), que son los encargados de definir el estándar de la estructura de las respuestas que el backend tendrá ante las solicitudes por parte del frontend.

Actividad 3.7
Se establecieron los servicios y sus correspondientes implementaciones, tanto para las reservas como para los puestos de trabajo, esto con el fin de manejar la data de la base de datos, tratarla y mandarla como respuesta.

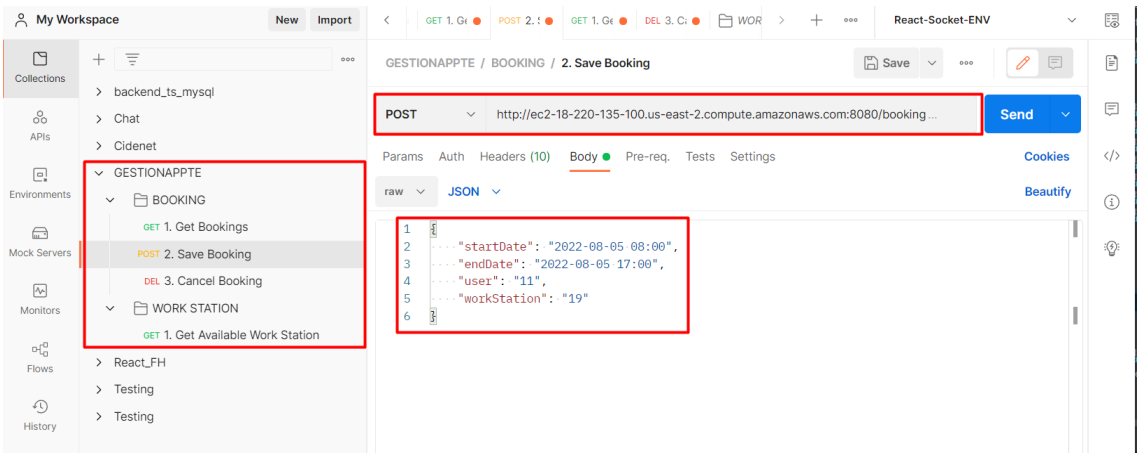
Actividad 3.8
Se describieron las diferentes Queries SQL encargadas de traer la información filtrada de la base de datos.

Actividad 3.9
Se realizó la conexión del backend con el servicio de la base de datos Amazon RDS.

Actividad 3.10
Para los aspectos relacionados con la seguridad de la aplicación, se decidió implementar buenas prácticas de desarrollo y seguridad de autenticación en el lado del backend. Para esto se implementa el protocolo de autenticación de Google OAuth2, uno de los más usados a nivel mundial y que provee un token de autenticación junto con datos específicos del cliente que está solicitando acceso a la plataforma.

Actividad 3.11
Se realizan pruebas desde el lado del backend con la herramienta de Postman.

Figura 13. Postman y estructura de pruebas funcionales



En la figura anterior, se muestra una petición POST en la herramienta de postman en su versión gratuita. Esta permite realizar peticiones a puntos de acceso locales y públicos.

5.4 Etapa 4: Conexión e implementación del frontend y backend.

Actividad 4.1
Se activó el frontend y el backend junto con su microservicio de forma local y en simultáneo.

Actividad 4.2
Se configuró el frontend con la direcciones URL locales de los diferentes endpoints con tal de comprobar interconexión, fiabilidad del funcionamiento, detección y solución de bugs(errores en la plataforma).

5.5 Etapa 5, Despliegue de la aplicación.

Actividad 5.1
Se desplegó el backend con el servicio en la nube de Amazon EC2, que permite generar instancias de servidores virtuales en alguna región del espectro geográfico de AWS. Aquí se expusieron las API's que se requerían. En el caso particular de Gestionappte fue una sola instancia que representaba el microservicio.

Actividad 5.2
Se configuró las variables de entorno del proyecto del frontend, esto para que las peticiones vayan dirigidas a la instancia EC2, siempre y cuando, esté en un entorno de producción, pues de lo contrario seguirá apuntando hacia los recursos locales de prueba.

Actividad 5.3
Se desplegó el frontend con el servicio en la nube de Amazon S3, que permite hospedar de manera estática páginas web, en este caso, una aplicación web de React Js.

Actividad 5.4
Se verificaron las funcionalidades mínimas del MVP.

5.6 Etapa 6: Validación con los líderes técnicos e integrantes del grupo de trabajo.

Actividad 6.1
Se revisaron todos los flujos de la aplicación web junto con líderes.

Actividad 6.2
Se puso a total disposición el código fuente del frontend y backend en los repositorios que tiene Gestionappte en los proyectos de Cidenet S.A.S dentro de la plataforma Gitlab.

Figura 14. Repositorio del frontend

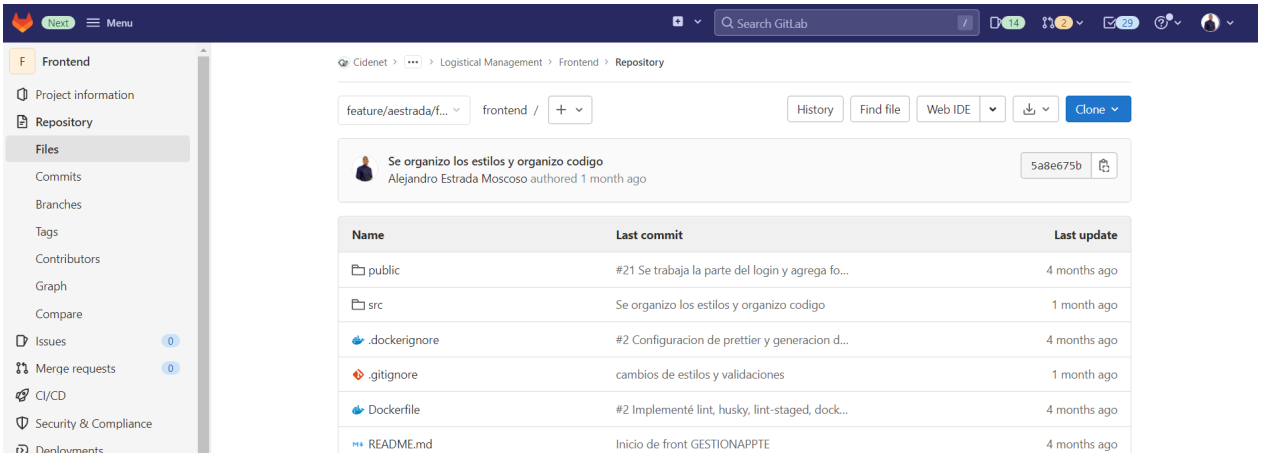
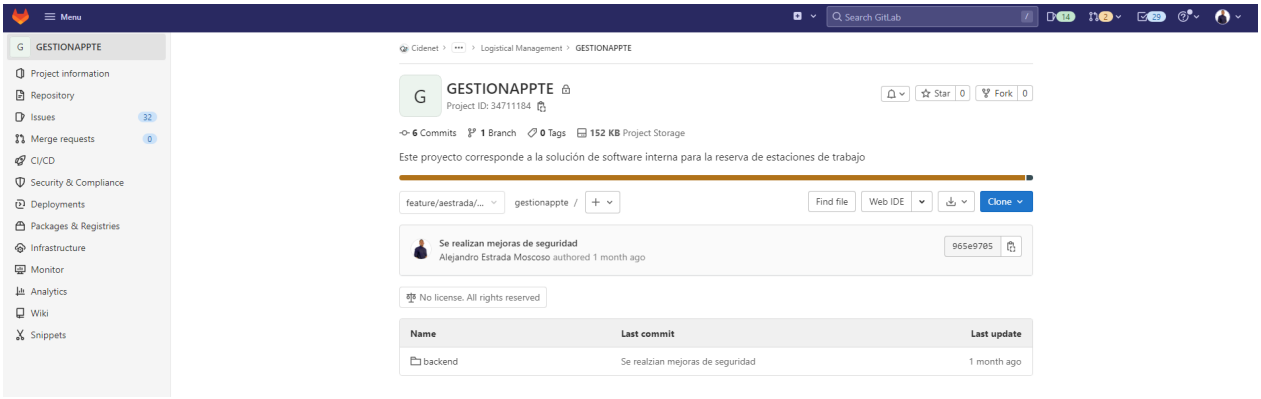


Figura 15. Repositorio del backend



Actividad 6.3
Se discutió con los líderes técnicos qué mejoras podría tener Gestionappte, así como los nuevos alcances visuales y funcionales que la aplicación experimentará en los próximos meses.

6. Resultados y Análisis

6.1 Contextualización

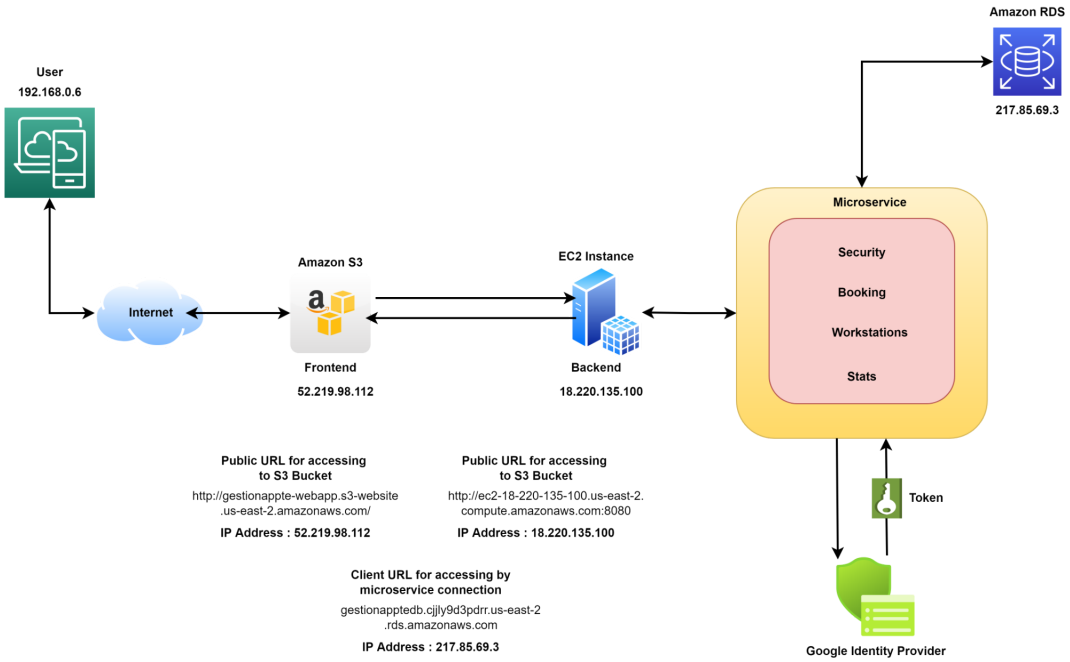
Para el desarrollo de la aplicación web responsive se tuvieron en cuenta ciertos factores relevantes como: el tipo de arquitectura, las estrategias de maquetación de los repositorios y la escalabilidad del proyecto en caso de que se decida a futuro aumentar su alcance hacia más sedes físicas de Cidenet S.A.S o incluso más funcionalidades internas.

En la parte del frontend, se decidió utilizar el framework de React Js, puesto que al ser uno de los marcos de trabajo más utilizados a nivel mundial, se sabía de su amplia biblioteca de librerías que podían servir al equipo para una adaptabilidad más corta en el momento de plasmar el desarrollo y estructuración de la parte visual del proyecto. Algunas de las herramientas que se utilizaron para llevar a cabo dicha codificación, fueron el editor de código Visual Studio Code, debido a su gran cantidad de extensiones que facilitan la programación. También se usaron algunos componentes Open Source como NVM (Node Version Manager), el cual permite gestionar la versión de Node, que básicamente brinda el entorno de desarrollo para poder renderizar líneas de código de Javascript y a su vez de React Js. Para la maquetación base se usó la interfaz de línea de comandos o más conocido CLI NPX, que permite facilitar la gestión e instalación de dependencias, con esto se obtuvo un esqueleto de React junto con los paquetes necesarios para empezar a desarrollar.

En el backend, se decidió usar el framework Java Spring Boot, debido a su amplia adaptabilidad al desarrollo de software empresarial, alta compatibilidad y con un amplio espectro de dependencias que facilitan el desarrollo de controladores y servicios. Además se hizo uso de algunos servicios en la nube, tanto para la autenticación, como para la consulta y edición de información en la base de datos. Se usó Maven como gestor de dependencias. En la parte de la codificación, se optó por el editor de código Intellij Idea en su versión “Community”, es decir, libre de costo.

A continuación, se muestra la arquitectura general de Gestionappte:

Figura 16. Estructura General de Gestionappte



En la figura 16, se observa la estructura del proyecto en alto nivel, que explica las conexiones entre los diferentes componentes de la aplicación. Al ser un diseño responsive, se desarrolló la aplicación web adaptable a una visión móvil. Esto debido a que el mayor uso de la plataforma por parte de los colaboradores de Cidenet S.A.S, será desde el smartphone personal.

Como se logra visualizar en el diagrama anterior, se tiene un entorno separado de desarrollo, es decir, el frontend separado del backend. Esto con el fin de evitar una arquitectura monolítica, aquella que en un mismo despliegue va integrado el frontend y el backend. Se pretende eludir esta arquitectura puesto que un fallo mínimo en el sistema puede acarrear múltiples errores en la aplicación. Si bien una arquitectura de microservicios puede presentar el mismo inconveniente, el fallo se centraliza en el microservicio defectuoso, es decir, no afecta el funcionamiento de los otros que componen el backend.

Tomando el término “servicio” en su sentido más estricto en la arquitectura de software, se refiere a la separación de funcionalidades de la aplicación, en este caso, frontend y backend. Tal como está representado en la figura 1. Pero en la práctica, la mayoría de fuentes hacen alusión a qué tan separados están los servicios que conforman el backend. En el caso particular de Gestionappte, se optó por un solo microservicio que se encargará de cuatro aspectos fundamentales:

1. Seguridad (Security)
2. Reservas (Bookings)
3. Estaciones de trabajo (Workstations)

4. Estadísticas (Stats)

Los motivos por los cuales se decidió trabajar con un solo microservicio se exponen a continuación: al ser un MVP (producto mínimo viable) que se encuentra en sus fases iniciales de implementación, tener la lógica de programación en más microservicios, podría aumentar el alcance mínimo del desarrollo, también acarrearía más costo por cada uno de los microservicios que se desplieguen en servicios en la nube como en este caso Amazon EC2. Este tipo de servicio que ofrece Amazon, cuenta con una capa de costo por tiempo en que la instancia esté activa y no por petición, es decir, el costo-beneficio no daba viabilidad de su separación.

Figura 17. Algunos precios de las instancias EC2

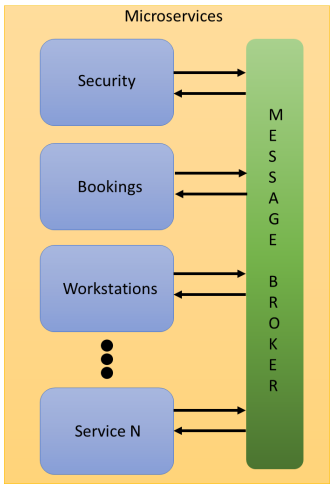
Nombre de la instancia	Tarifa por hora bajo demanda	vCPU	Memoria	Almacenamiento	Rendimiento de la red
a1.medium	0,0255 USD	1	2 GiB	Solo EBS	Hasta 10 gigabits
a1.large	0,051 USD	2	4 GiB	Solo EBS	Hasta 10 gigabits
a1.xlarge	0,102 USD	4	8 GiB	Solo EBS	Hasta 10 gigabits
a1.2xlarge	0,204 USD	8	16 GiB	Solo EBS	Hasta 10 gigabits
a1.4xlarge	0,408 USD	16	32 GiB	Solo EBS	Hasta 10 gigabits
a1.metal	0,408 USD	16	32 GiB	Solo EBS	Hasta 10 gigabits
t4g.nano	0,0042 USD	2	0,5 GiB	Solo EBS	Hasta 5 gigabits
t4g.micro	0,0084 USD	2	1 GiB	Solo EBS	Hasta 5 gigabits
t4g.small	0,0168 USD	2	2 GiB	Solo EBS	Hasta 5 gigabits

Fuente: Amazon Web Service. Precios de las instancias bajo demanda. [Consultado: 05 de Septiembre de 2022]. Disponible en: https://aws.amazon.com/es/ec2/pricing/on-demand/

Cabe aclarar que un microservicio no siempre se encarga de un solo servicio para la aplicación, esto es meramente ideal. En muchos casos, se separan las responsabilidades del backend en diferentes microservicios que a su vez internamente se encargan de más de un servicio. Por ejemplo, un microservicio que se encargue de las reservas, puede que de manera transversal tenga el rol de alimentar con datos estadísticos la asistencia de personal hacia otros microservicios o hacia otros usuarios finales.

Finalmente, al trabajar con una arquitectura de microservicios, se obtiene el beneficio de tener una estructura escalable puesto que se puede aumentar la cantidad de servicios dependiendo de las exigencias y necesidades del proyecto o por las necesidades que tenga el Product Owner en el momento de diseñar el sistema.

Figura 18. Estructura Escalable de una arquitectura de microservicios orientada al proyecto Gesionappte



En la figura anterior, se describe el diseño general del backend, que cuenta con los microservicios fundamentales como: seguridad, reservas y estaciones de trabajo. También se evidencia, que es un sistema adaptable, es decir, capaz de recibir más conexiones de diferentes microservicios. En caso de que se necesite la comunicación entre puntos de acceso, se usaría un message broker que se encargaría de generar un cola de peticiones entre servidores y cada vez que el servidor responde una solicitud, va despejando la cola de peticiones ya sea de agregar, editar, leer o eliminar. Uno de los message broker open source más usados es RabbitMQ, el cual se adaptaría perfectamente al proyecto.

6.2 Implementación

Para la implementación de Gestionappte, se definieron las funcionalidades fundamentales que esta cumpliría cuando fuera desplegada para el uso público de los colaboradores de Cidenet S.A.S.

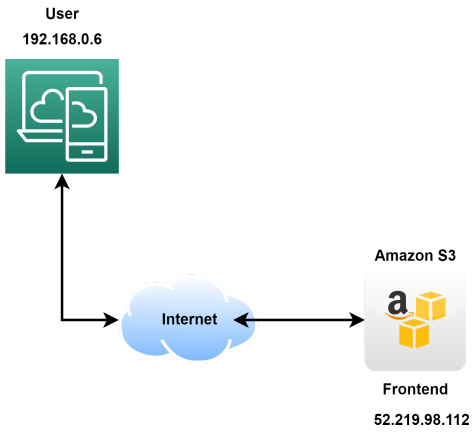
Tabla 5. Alcance de funcionalidades del usuario

#	Funcionalidad
1	Iniciar Sesión
2	Consultar estaciones disponibles
3	Reservar estaciones
4	Confirmar reserva
5	Cancelar reserva
6	Recibir notificaciones
7	Consultar estadisticas de asistencia

A continuación se definirán las etapas técnicas del desarrollo frontend y backend:

Frontend

Figura 19. Frontend Gestionappte



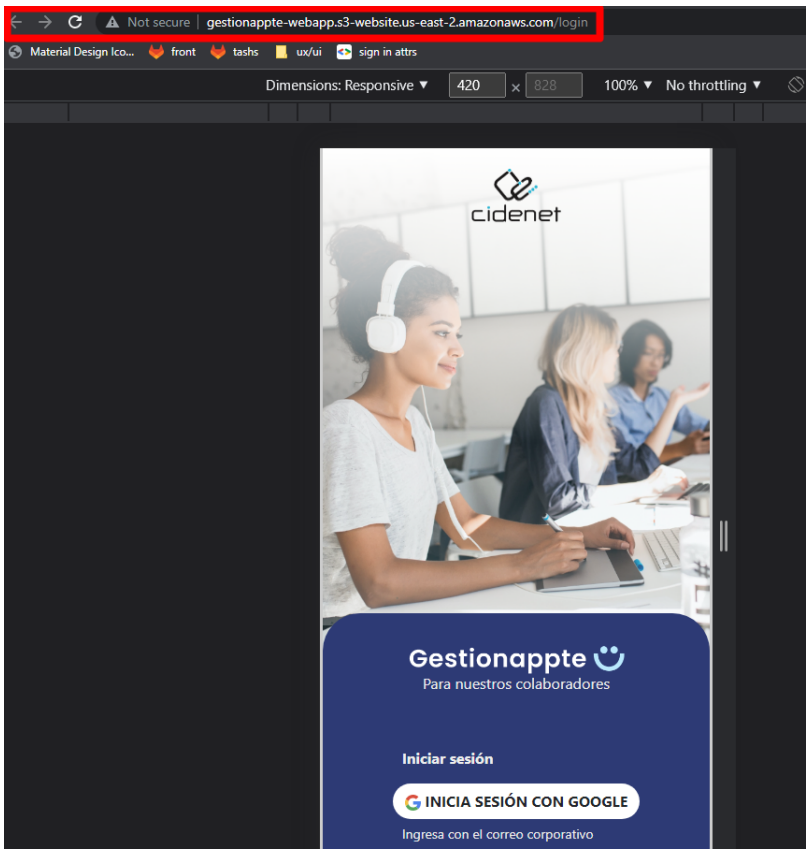
En la anterior figura, se tiene el esquema específico del lado del frontend donde se evidencian las conexiones de los usuarios finales, ósea, de los colaboradores de Cidenet S.A.S y cómo estos interactúan con la plataforma Gestionappte. A continuación se describen aspectos claves que hicieron posible su integración.

Alojamiento de los recursos visuales de React Js

Para dicha tarea se usó el servicio en la nube de Amazon S3, donde su unidad fundamental son los buckets, es decir, contenedores de objetos que almacenan información de cualquier tipo. Estos objetos pueden ir desde imágenes y archivos de cualquier índole, hasta recursos estáticos como los de páginas web, en este caso el frontend de React.

Entre los diferentes tipos de almacenamiento configurables para los buckets de S3, se optó por la capa de almacenamiento Estándar S3, pues brinda alta durabilidad, disponibilidad, rendimiento y baja latencia en el momento de solicitar los objetos.

Figura 20. Despliegue del frontend en S3



En la figura anterior, se muestra el bucket s3 que almacena todos los componentes estáticos de la página web, permitiendo así, su visualización e interacción.

Es importante mencionar que los objetos o recursos del frontend son repartidos en mínimo tres zonas de disponibilidad distintas, es decir, en diferentes dispositivos(servidores) que se encuentran cada uno separados por kilómetros en una región de AWS (Ubicación geográfica que contiene varias zonas de disponibilidad). Esto con el fin de tener disponibilidad de la información en caso de que una zona falle por algún motivo técnico. En el caso de Gestionappte se configuró el bucket S3 en la región US-EAST-2. Normalmente, se trata de configurar las regiones más cercanas al punto donde las conexiones se originan, en este caso particular, en Colombia. Aunque la región más cercana está en Brasil, Sao Paulo, se eligió la de Estados Unidos, por motivos de soporte y fiabilidad.

Diseño responsive

Para poder llevar a cabo este tipo de tarea, se implementaron “media queries” que sirven para cambiar el estilo CSS del frontend a medida que la resolución de la pantalla disminuye, es decir, cuando diferentes tipos de smartphones tratan de acceder a Gestionappte. También en ciertas librerías de diseño como Material UI, se utilizaron estrategias nativas de ellas mismas para lograr esta adaptabilidad.

Figura 21. Ejemplos de media queries usados en el frontend

```
@media (max-width: 414px) {
    .login-body {
        position: relative;
        height: 100vh;
        width: 100%;
    }
}

@media (max-width: 380px) {
    .bottom-card {
        padding: 10px;
        padding-top: 30px;
    }

    .name-container {
        width: 90%;
    }

}
```

Figura 22. Ejemplos de la manera nativa en que material ui implementa el responsive usados en el frontend

```
const useStyles = makeStyles((theme) => ({
    modal: {
        position: 'absolute',
        width: 500,
        backgroundColor: 'white',
        border: '2px solid',
        borderRadius: '20px',
        boxShadow: theme.shadows[5],
        padding: theme.spacing(2, 4, 3, 4),
        top: '50%',
        left: '50%',
        [theme.breakpoints.down(500)]: {
            width: 412,
        },
        [theme.breakpoints.down(416)]: {
            width: 388,
        },
        [theme.breakpoints.down(388)]: {
            width: 360,
        },
        [theme.breakpoints.down(360)]: {
            width: 280,
        },
    },
```

En las dos figuras anteriores, se describen algunos ejemplos claros de la implementación responsive dentro de la aplicación. En la primera, se exponen media queries, estas se encargan de acoplar el componente que cuente con dicho estilo a la resolución. En la segunda, se muestran breakpoints o puntos de quiebre que usa la librería de material UI, para modificar las dimensiones de sus componentes en relación con las dimensiones de la pantalla.

Paquetes de node para el uso de librerías en el proyecto

Gracias a la vasta comunidad que tiene React Js, se ha construido una base sólida de paquetes que contienen funcionalidades que facilitan la experiencia del desarrollador y acortan el tiempo en la construcción de componentes que a primera instancia pueden ser algo complejos. En Gestionappte se usaron algunos paquetes que simplificaron los pasos de las construcciones de las vistas de la aplicación, algunos de los paquetes más relevantes usados fueron:

1. material ui (cuenta con una amplia gama de componentes estilizados como modales, time picker, date pickers, etc…)
2. axios (cliente HTTP que facilita el envío y recepción de peticiones)
3. chart-js (permite crear componentes funcionalidades con tablas y gráficas)
4. sweet alert (brinda mensajes de alerta)
5. prettier (organiza y limpia el código)
6. jwt-decode (sirve para decodificar el token y capturar la información que este lleva)

Todos estos paquetes se encuentran en el archivo package.json, este almacena información de comandos para el renderizado y testing de la aplicación, así como los paquetes usados junto a sus versiones.

Figura 23. Parte del archivo package.json de Gestionappte

Token

Para ingresar a la plataforma, se provee un token que entrega el backend, cuando se confirman los datos de ingreso. Este token es almacenado por parte del frontend en el localstorage del navegador, sirve para obtener datos superficiales del usuario como el nombre e identificación dentro de la plataforma, es decir, valores que no puedan ser usados para violentar los recursos de Gestionappte.

Figura 24. Localstorage del navegador almacenando el token

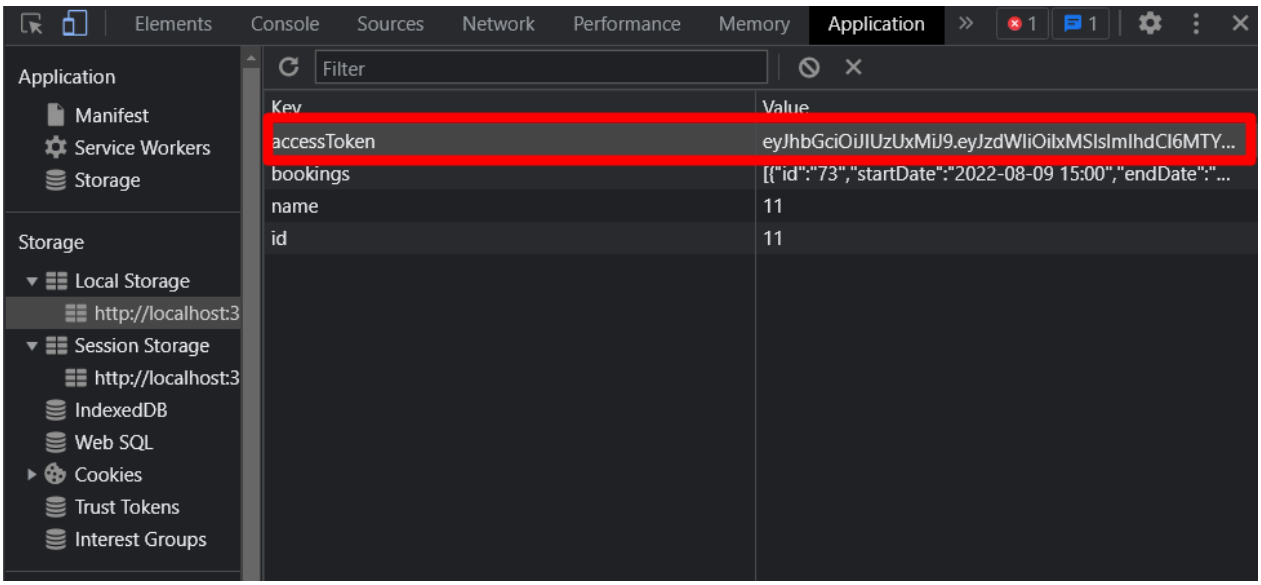


Figura 25. Token decodificado

Como se observa en la imagen anterior, el token es fácil de decodificar, existen páginas como "jwt.io" que decodifican la mayoría de tokens.

Rutas Privadas y Públicas

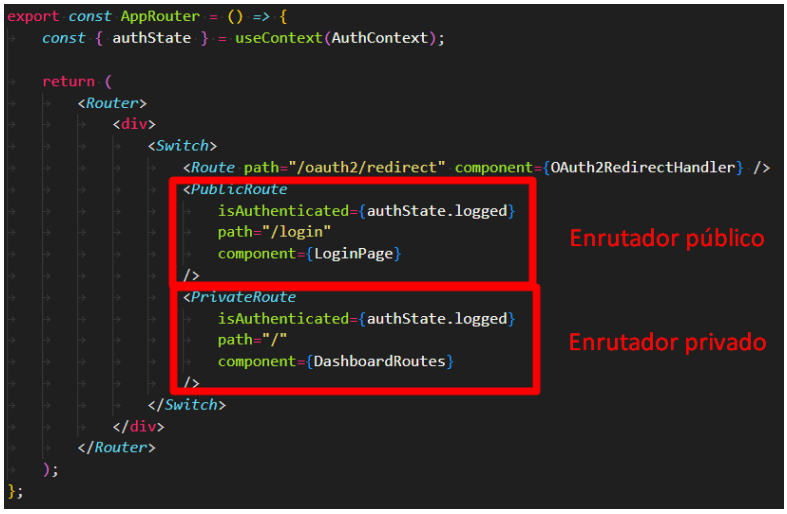
Esta sección es importante de recalcar pues integra la experiencia de usuario en términos de seguridad, es decir, Gestionappte válida cuando un usuario ingresa a la plataforma de manera correcta, en ese caso, rutas públicas como el login no le son posibles de acceder a menos que el usuario cierre sesión y viceversa, cuando el usuario no ha ingresado sesión, el deber ser es que la plataforma le bloquee el acceso a rutas privadas como la creación de reservas o consulta de estadísticas. Para llevar a cabo esta tarea, se usó un reducer de React que se encarga de guardar el

estado global de la sesión y luego con un validador en el router, este se encarga de mostrar o no las rutas de la aplicación.

Figura 26. Reducer usado para el estado global de la sesión del usuario

```
import { types } from "../../constants/types";

export const authReducer = (state, action) => {
    switch (action.type) {
        case types.login:
            return {
                ...state,
                logged: true,
                name: action.payload.name,
            };

        case types.logout:
            return {
                ...state,
                logged: false,
                name: "",
            };
        You, 4 months ago • #Implementacion logica
        default:
            return state;
    }
};
```

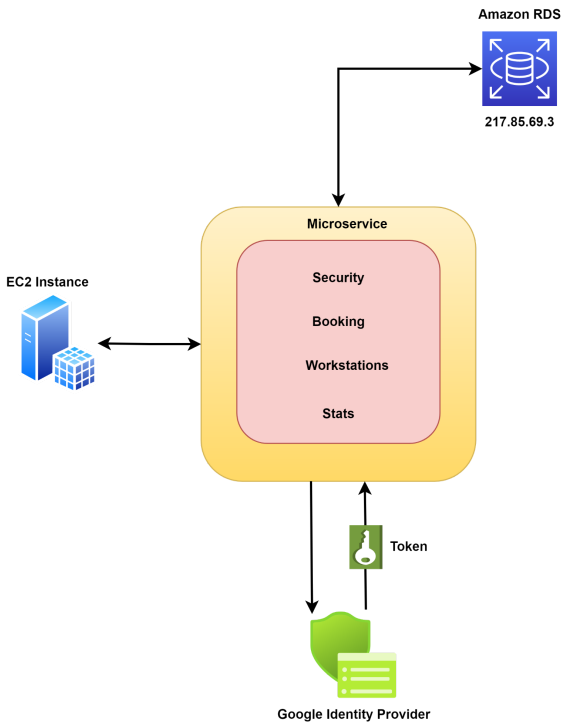
Figura 27. Enrutador de las rutas públicas y privadas




```
export const AppRouter = () => {
    const { authState } = useContext(AuthContext);

    return (
        <Router>
            <div>
                <Switch>
                    <Route path="/oauth2/redirect" component={OAuth2RedirectHandler} />
                    <PublicRoute
                        isAuthenticated={authState.logged}
                        path="/login"
                        component={LoginPage}
                    />
                    <PrivateRoute
                        isAuthenticated={authState.logged}
                        path="/"
                        component={DashboardRoutes}
                    />
                </Switch>
            </div>
        </Router>
    );
};
```

Backend

Figura 28. Backend Gestionappte



Alojamiento del microservicio

En la implementación del servicio en la nube para desplegar el microservicio, se decidió optar por el servicio de Amazon EC2 (Elastic Compute Cloud), servicio que permite configurar servidores virtuales con la información necesaria para su funcionamiento. Para este servicio existen diferentes capas de cobro, en Gestionappte se usó el cobro por tiempo de actividad de la instancia, el más sencillo de todos, con tal de tener el microservicio en la nube de manera rápida, con alta disponibilidad y seguridad. A futuro, se piensa poner alojar este microservicio en los servidores propios de Cidenet S.A.S con tal de evitar grandes costos.

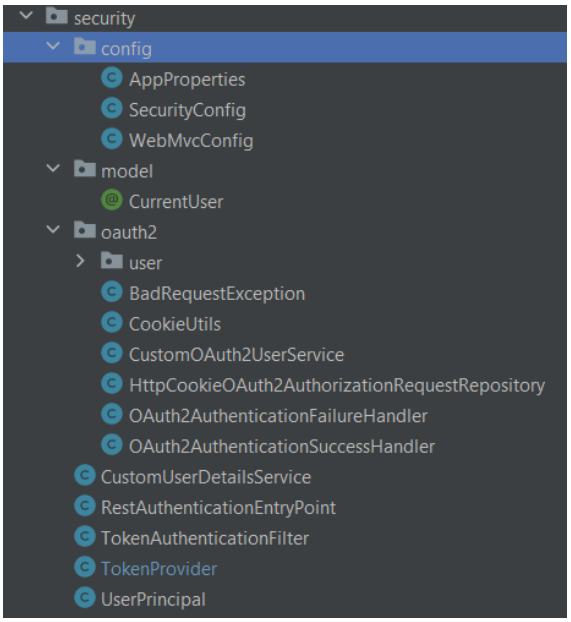
Figura 29. Despliegue del backend

```
docker push 935430155669.dkr.ecr.us-east-2.amazonaws.com/gestionappte:latest
The push refers to repository [935430155669.dkr.ecr.us-east-2.amazonaws.com/gest
ionappte]
cca738e4bafe: Preparing
6019de34807c: Preparing
208a168f180d: Preparing
601bff960b80: Preparing
f199ad476e12: Preparing
631743b7098c: Preparing
631743b7098c: Waiting
f199ad476e12: Layer already exists
601bff960b80: Layer already exists
631743b7098c: Layer already exists
6019de34807c: Pushed
cca738e4bafe: Pushed
208a168f180d: Pushed
latest: digest: sha256:da93edd05d28525261480c95dd50ce82e4ccb5944802c29b81c8aaa00
28661fd size: 1577
```

Seguridad

En la parte de la seguridad, se implementó el servicio que brinda GCP(google cloud platform) de autenticación por parte de google, Google OAuth2 . Se decidió este modelo, debido a que Cidenet S.A.S tiene contratados los servicios de google workspace, por lo que los correos de los colaboradores están asociados a google. Entonces lo que se hizo fue permitir el acceso solo para los correos de dominio @cidenet.com.co, diferente a este son bloqueados y no dan un token de respuesta para el acceso a Gestionappte.

Figura 30. Seguridad con google OAuth2 en el backend



Reservas

Para la parte de las reservas, se desarrollaron 4 endpoints(puntos de acceso en el back), que controlan:

Tabla 6. Tipos de endpoints

#	Tipo de endpoint
1	Listado de reservas
2	Guardar reservas
3	Confirmar/Cancelar reservas
4	Consulta de estadisiticas

Figura 31. Endpoint listado de reservas

```
@GetMapping
public Object getBookings(Authentication authentication) throws Exception {

    List<Booking> bookings = this.bookingService.getBookings(this.getUserId(authentication));

    List<BookingDTO> bookinDTOs = bookings.stream()
            .map(b -> modelMapper.map(b, BookingDTO.class)).collect(Collectors.toList());

    return ok(bookinDTOs);
}
```

Figura 32. Endpoint guardar Reservas

```
@PostMapping
public Object saveBooking(Authentication authentication, @RequestBody BookingDTO bookingDTO)
        throws Exception {
    Booking bookingToSave = modelMapper.map(bookingDTO, Booking.class);
    bookingToSave.setUser(new User(this.getUserId(authentication)));
    bookingToSave.setWorkStation(new WorkStation(bookingDTO.getWorkStation()));

    bookingToSave.setStatus("CREATED");

    Booking booking = this.bookingService.saveBooking(bookingToSave);

    return ok(modelMapper.map(booking, BookingDTO.class));
}
```

Figura 33. Endpoint confirmar/cancelar reservas

```
@GetMapping("/status/{id}/{status}")
public Object confirmBookings(@Valid @PathVariable Long id, @Valid @PathVariable String status)
        throws Exception {

    String bookings = this.bookingService.updateBooking(id, status);

    return ok(bookings);
}
```

Figura 34. Endpoint consulta de estadísticas

```
@GetMapping("/stats-by-attendance")
public Object getStatsByAttendance(Authentication authentication,
                                   @RequestParam String reportType,
                                   @RequestParam String datePicker,
                                   @RequestParam String radioValue,
                                   @RequestParam Long selectedDateInit,
                                   @RequestParam Long selectedDateEnd
                                   ) throws Exception {
    List<Booking> listOfBookings;
```

Estaciones de trabajo

Para la parte de las estaciones de trabajo, se desarrolló un endpoint que verifica en la base de datos cuáles puestos de trabajos están disponibles en los rangos de inicio y fin de la reserva en el momento de llenar el formulario de reservas. Esto para filtrar cuáles puestos ya están ocupados y evitar inconvenientes con las asignaciones de puestos por parte de la plataforma.

Figura 35. Endpoint listado de puestos de trabajo disponibles

```
@GetMapping
public Object getAvailableWorkstations(@Valid @RequestParam String startDate,
        @Valid @RequestParam String endDate, @Valid @RequestParam String type)
        throws Exception {

    Date startDateParsed = DateUtils.parseDateStrictly(startDate, ...parsePatterns: "yyyy-MM-dd HH:mm");
    Date endDateParsed = DateUtils.parseDateStrictly(endDate, ...parsePatterns: "yyyy-MM-dd HH:mm");

    modelMapper.typeMap(WorkStation.class, WorkStationDTO.class).addMappings(mapper -> {
        mapper.map(src -> src.getFloor().getId(), WorkStationDTO::setFloor);
    });

    List<WorkStationDTO> workStationDTOs = this.workStationService
            .getAvailableWorkstations(startDateParsed, endDateParsed, type).stream() Stream<WorkStation>
            .map(user -> modelMapper.map(user, WorkStationDTO.class)) Stream<WorkStationDTO>
            .collect(Collectors.toList());

    return ok(workStationDTOs);
}
```

Notificaciones

Para la parte de las notificaciones, Gestionappte emplea un envío de correo avisando a las personas que tienen sus reservas en estado “creada”, que confirmen la reserva; en caso de que no lo hagan, esta libera el espacio para que sea ocupado por otra persona. Los correos de notificación se envían 24 horas antes, se vuelve a enviar un correo una hora antes de la reserva. Finalmente, si en 10 minutos antes de la reserva no se confirma, Gestioanppte libera el espacio de manera automática.

Para lograr esto, se implementó un cron job (ejecución automática de tareas), que se ejecuta en todo momento y verifica los tiempos de las reservas con la hora actual, las compara y dependiendo de las validaciones ejecuta una acción, que puede ser enviar un correo o cancelar la reserva. En el envío de correos se utilizó el protocolo de transferencia de correo con gmail SMTP que integra java spring boot.

Figura 36. Cron usado en el envío de correo o cancelación de reservas

```java
@Async
@Override
@Scheduled(cron = "0 */5 * * * *")
public void lookout() {
    Date now = Calendar.getInstance().getTime();
    List<Booking> bookings = this.bookingRepository.findBookingToExpired(now);

    bookings.stream().forEach(b -> {
        long diffInMillies = Math.abs(now.getTime() - b.getStartDate().getTime());
        long diff = TimeUnit.MINUTES.convert(diffInMillies, TimeUnit.MILLISECONDS);
        long diffDay = TimeUnit.HOURS.convert(diffInMillies, TimeUnit.MILLISECONDS);

        if (diffDay > 1 && diffDay <= 24 && !"NOTIFIED_DAY".equalsIgnoreCase(b.getStatus())) {
            // enviar correo de recordario y marcar como notificado dia
            this.notifyBookingStatus(b.getUser(), b, this.getTemplate( type: "NOTIFIED_DAY", b),
                    status: "NOTIFIED_DAY");
        } else if (diff <= 60 && !"NOTIFIED_HOUR".equalsIgnoreCase(b.getStatus())) {
            // enviar correo y marcar como notificado
            this.notifyBookingStatus(b.getUser(), b, this.getTemplate( type: "NOTIFIED_HOUR", b),
                    status: "NOTIFIED_HOUR");
        } else if (diff <= 10) {
            // se libera el puesto
            b.setStatus("CANCELLED");
            this.bookingRepository.save(b);
        }
    });
```

Base de datos

En esta sección, una de las partes principales del backend , se dispuso de una instancia de base de datos desplegada en la nube de AWS, haciendo uso del servicio Amazon RDS (Relational Database Service). En este apartado, se configuró la base de datos con el motor de SQL, PostgreSQL, que es un sistema de base de datos relacional orientado a objetos de código abierto con una sólida reputación de fiabilidad, estabilidad y precisión. Al ser la primera versión del MVP, se trabaja con la capa gratuita de almacenamiento con una duración de dos meses. A medida que la instancia empieza a guardar más registros, se tiene pensado cambiar la capa de cobro en una que no sea tan costosa y bajo el asesoramiento de AWS.

El backend se conecta directamente con la base de datos, y mediante queries SQL, se extrae la información pertinente, teniendo en cuenta los requerimientos de la petición. También es importante recalcar que al ser configurada para el acceso público, esta cuenta con una IP pública que permite, mediante un cliente SQL, acceder a la base de datos de manera remota y modificar al antojo del cliente y sin intermediarios (en este caso el backend). Esta última posibilidad fue crucial en la etapa de testing donde se necesitaban introducir datos de prueba para poderlos visualizar en la plataforma.

Figura 37. Estructura de los módulos del backend y sus conexiones con la base de datos

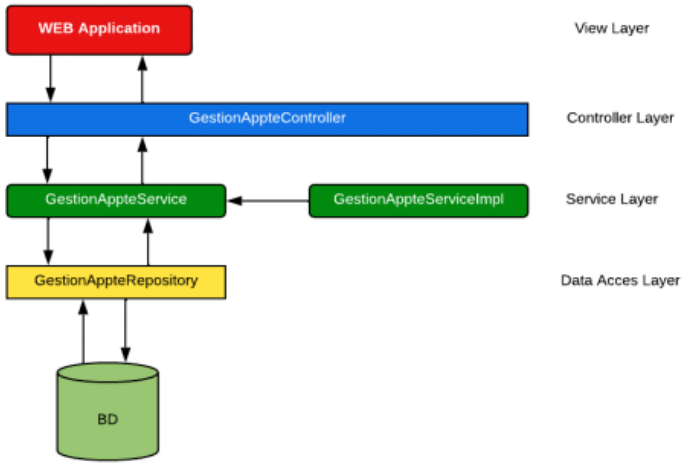


Figura 38. Configuración del cliente SQLElectron con la instancia RDS

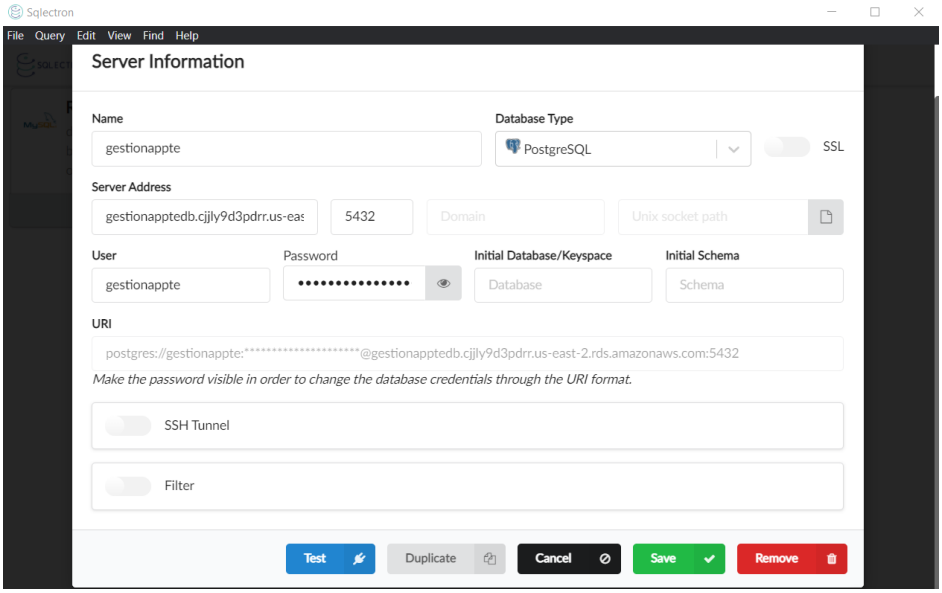
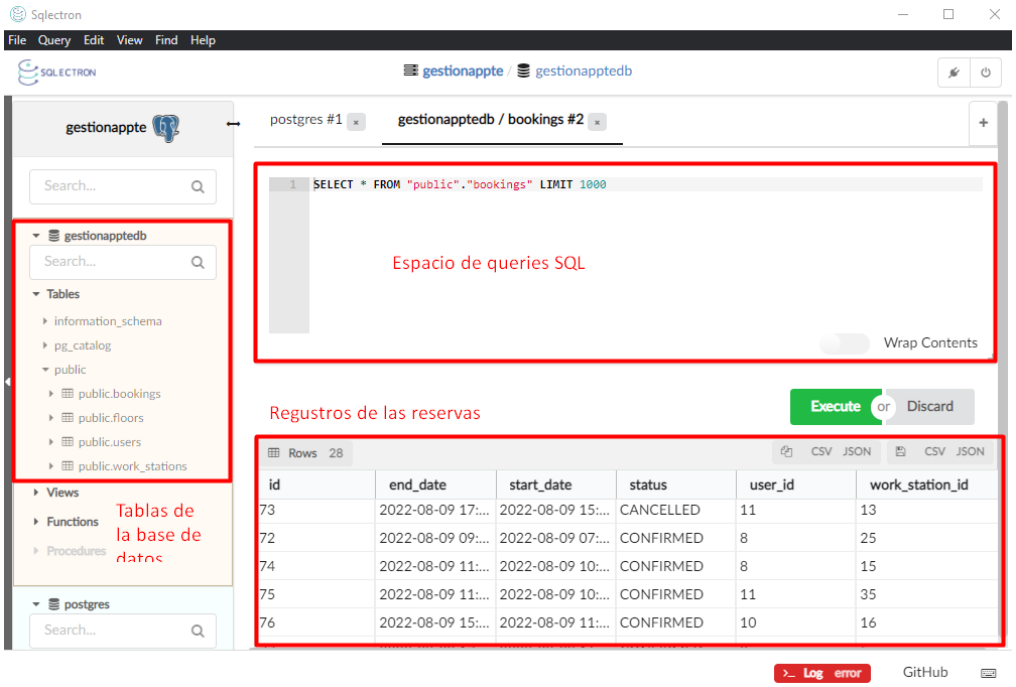


Figura 39. Estructura de la base de datos en el cliente SQLElectron

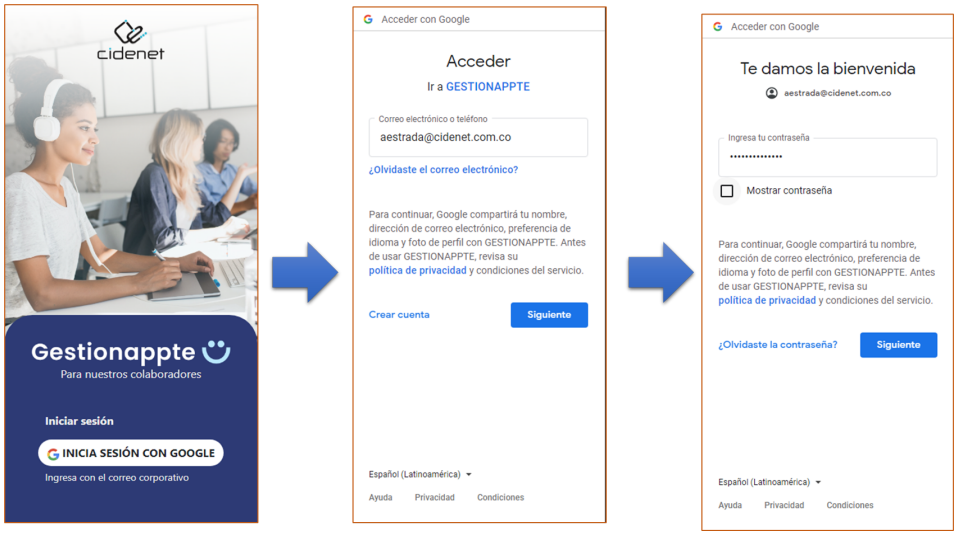


6.3 Resultados

A continuación, se expondrán los diferentes módulos y vistas de Gestionappte, así como las funcionalidades y acciones que el cliente actualmente puede realizar con el MVP. Las siguientes imágenes corresponden a la visualización móvil, en los anexos se encuentran las mismas vistas pero con la resolución web.

6.3.1 Login

Figura 40. Paso a paso del login en Gestionappte



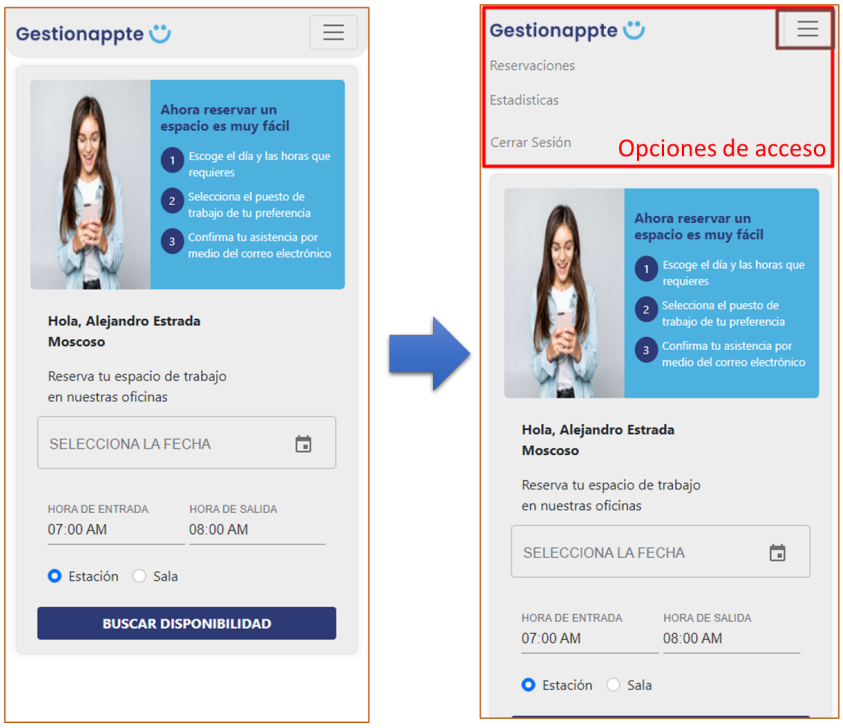
En la figura anterior, se representan las diferentes visitas en su versión móvil, donde se tienen los diferentes pasos que un usuario con el correo de dominio @cidenet.com.co puede iniciar sesión, solo basta con ingresar las credenciales del correo y autorizar a Gestionappte para el uso de datos del cliente. Si el usuario no está registrado en la plataforma, se creará un usuario en la base de datos con un identificador único, que servirá como punto de entrada en la trazabilidad de las reservas y estadísticas.

Tabla 7. Pasos secuenciales de las acciones del cliente para el acceso a Gestionappte

#	Pasos del Login
1	Ingresar a Gestionappte
2	Darle click al botón de inicio de sesión por google
3	Escribir el correo con dominio de Cidenet
4	Digitar la contraseña

6.3.2 Home

Figura 41. Home principal de Gestionappte



En la figura anterior, se observa el home principal de Gestionappte, es decir, la primera vista que tiene el cliente por defecto, al ingresar a la plataforma. En la parte superior derecha se encuentra un botón que despliega las opciones de acceso hacia las otras funcionalidades: Reservaciones,

Estadísticas y cerrar sesión. También en el home, se encuentra el formulario que el usuario debe diligenciar para consultar los puestos de trabajo disponibles.

6.3.3 Flujo en la creación de reservas

Figura 42. Formulario de reserva

En la imagen anterior, se visualiza el formulario que el colaborador debe diligenciar justo antes de buscar la disponibilidad de espacios de trabajo, pues con estos datos se filtran las coincidencias y se mostrarían los espacios totalmente disponibles para su elección.
El formulario consta de:

1. Datepicker (espacio para escoger fecha de la reserva)
2. Horas de entrada y salida de la reserva
3. Tipo de puestos de trabajo (estación o sala)

Se procede a escoger la fecha y los horarios.

Figura 43. Elección de la fecha y rangos horarios

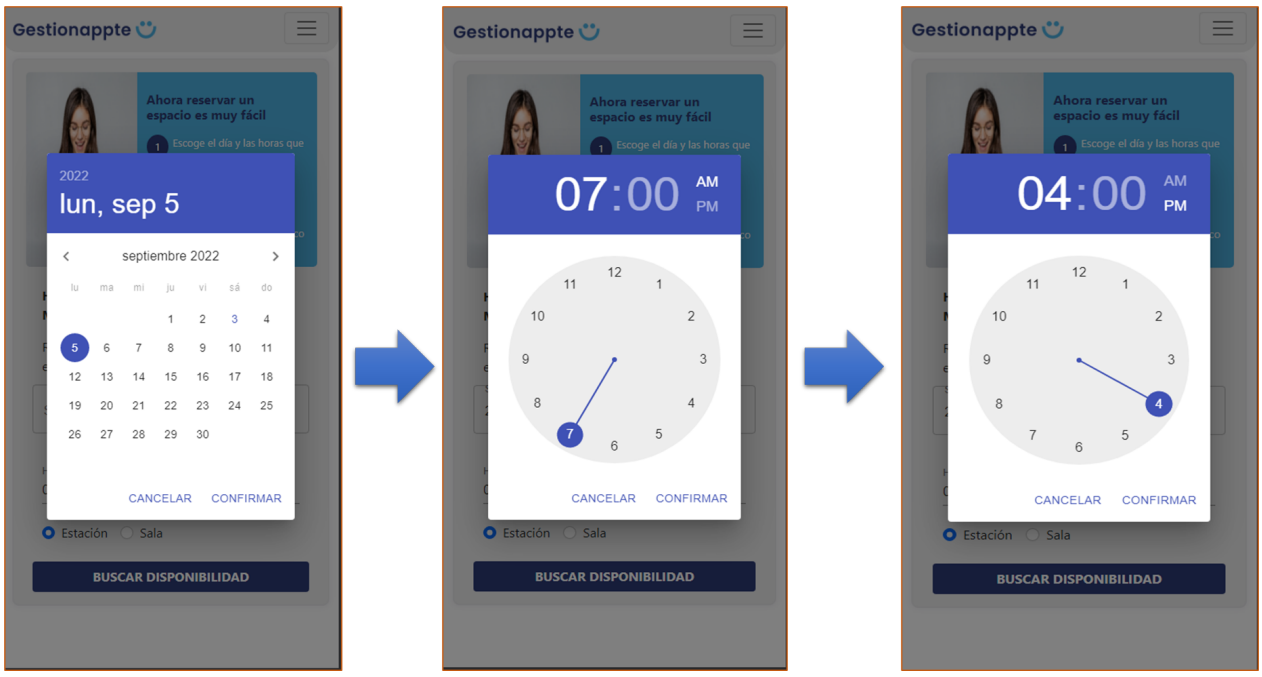


Tabla 8. Pasos secuenciales de las acciones del cliente para diligenciar el formulario

#	Pasos del formulario de reserva
1	Click en "Seleccion la fecha"
2	Escoger el día
3	Click en "HORA DE ENTRADA"
4	Elegir la hora de entrada, entre 7:00 - 19:00
5	Click en "HORA DE SALIDA"
6	Elegir la hora de salida, entre 7:00 - 19:00
7	Escoger el tipo de espacio de trabajo
8	Click en "BUSCAR DISPONIBILIDAD"

Después de consultar las disponibilidades, se despliega una modal que brinda una lista de espacios de trabajo disponibles filtrados dependiendo las entradas del formulario.

Figura 44. Despliegue de modal y selección de puestos de trabajo

Una vez se despliegue la modal, se selecciona el puesto de trabajo de preferencia, gracias a una lista con los resultados que entrega el backend, todos filtrados, garantizando fecha de agendamiento así como hora de entrada y de salida.

Tabla 9. Pasos secuenciales en la modal de creación de reservas

#	Pasos en la modal de selección de puestos de trabajo
1	Click en "Seleccione el puesto de trabajo"
2	Eligir el puesto de trabajo de preferencia
3	Click en "CREAR RESERVA" una vez este se active

6.3.4 Listado de reservaciones

En esta sección, se visualiza la ruta donde el usuario puede ver su historial de reservas creadas, confirmadas y canceladas. También se cuenta con una barra de búsqueda que permite filtrar las reservas por estado o por número de reserva.

Figura 45. Listado de reservas asociadas al usuario

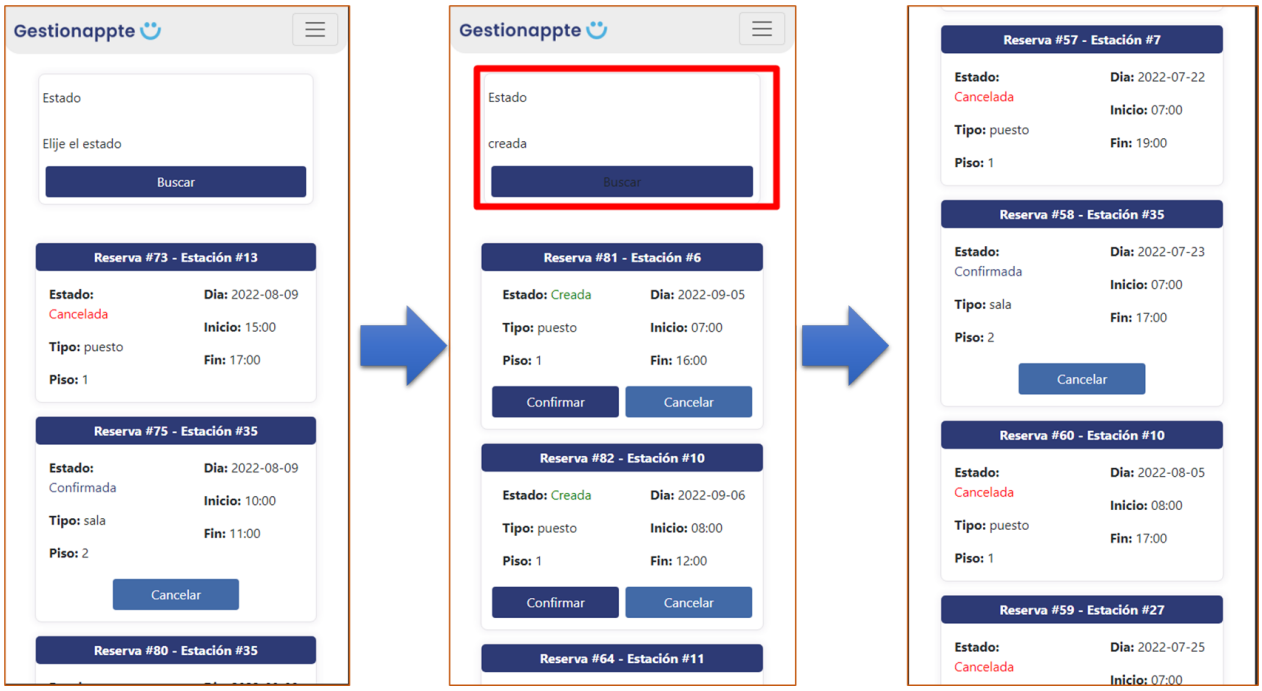


Tabla 10. Pasos secuenciales para el acceso al listado de reservas

#	Pasos para el acceso al listado de reservas
1	Desde el home, click en las opciones
2	Seleccionar "Reservaciones"

En cada reserva que muestra el listado, se encuentran dos botones que se encargan de confirmar o cancelar la reserva de manera directa en Gestionappte, sin la necesidad de hacerlo mediante los correos que envía la misma plataforma. Una vez la reserva es confirmada, los correos no se envían mas.

Figura 46. Confirmación de reserva en Gestionappte

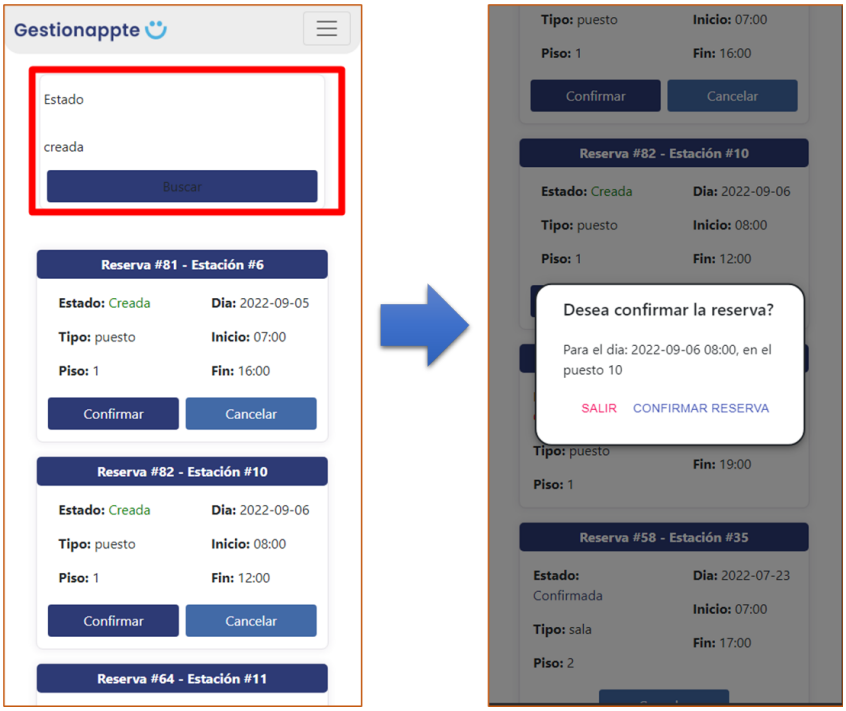


Tabla 11. Pasos secuenciales para la confirmación de reservas

#	Pasos para la confirmación de reservas
1	Desde el listado de reservas, elegir una reserva que tenga el boton de confirmar y darle click
2	En la modal, darle click en "CONFIRMAR RESERVA"

Figura 47. Cancelación de reserva en Gestionappte

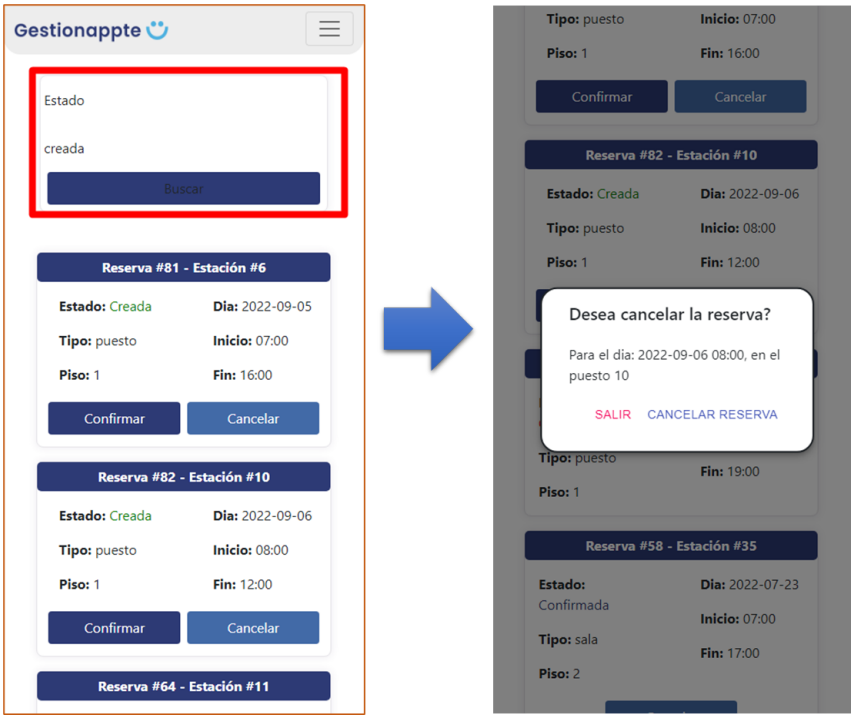


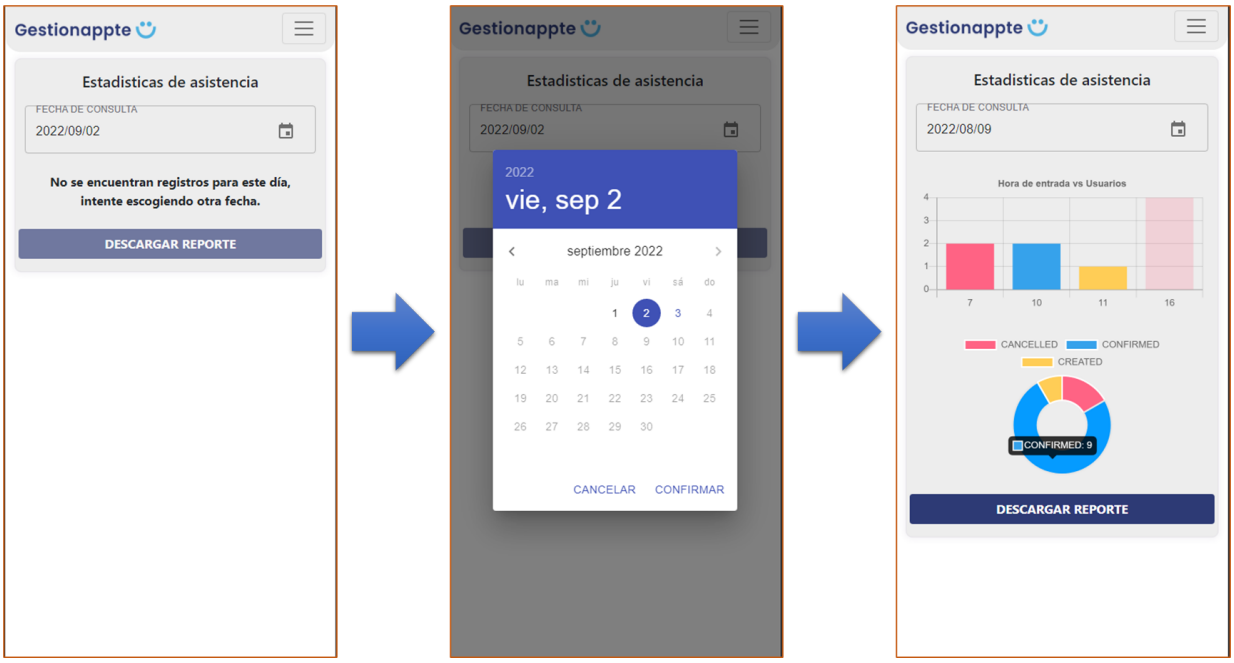
Tabla 12. Pasos secuenciales para la cancelación de reservas

#	Pasos para la cancelación de reservas
1	Desde el listado de reservas, elegir una reserva que tenga el boton de cancelar y darle click
2	En la modal, darle click en "CANCELAR RESERVA"

6.3.5 Estadísticas

En este apartado, se encuentran las vistas de la sección de las consultas estadísticas de asistencia, que servirán para el control y la trazabilidad de asistencia en el momento en que un administrador necesite visualizar la cantidad de colaboradores que fueron a la sede en un día en específico y en un rango horario determinado.

Figura 48. Estadísticas de asistencia



En la figura anterior, se ilustra la sección de Gestionappte donde el usuario-administrador podrá observar gráficas dicientes sobre:

1. Gráfica de hora de entrada de los colaboradores vs cantidad de personas
2. Gráfica en forma de pastel donde se visualiza el estado de las reservas

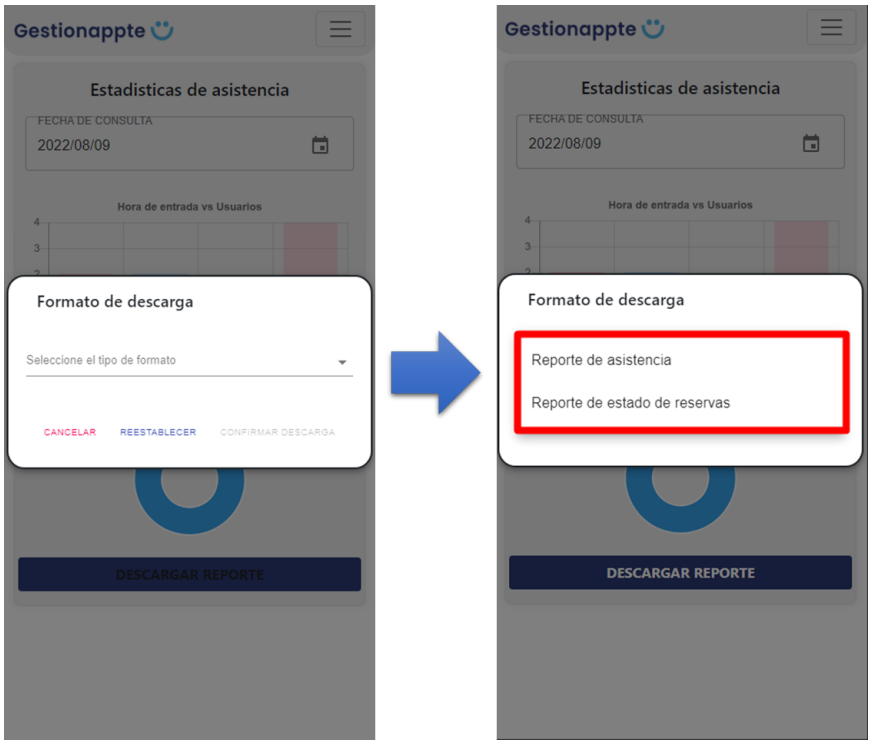
Al inicio, la vista de estadísticas muestra de manera automática las gráficas del día actual, pero también tiene un datepicker que permite consultar información sobre fechas previas.

Tabla 13. Pasos secuenciales para el acceso a las estadísticas de asistencia

#	Pasos para el acceso a las estadisticas de asistencia
1	Desde el home, click en las opciones
2	Seleccionar "Estadisticas"

Una vez, se haya seleccionado el día y ese día tenga datos, osea aparezcan gráficas. Es posible descargar un reporte de asistencia en formato excel con ciertos filtros que el usuario-administrador puede escoger, con el fin de tener información filtrada y útil.

Figura 49. Modal para la elección de filtros del reporte



Al desplegarse la modal, se tiene un listado de opciones que servirán para el formato de la información en la descarga:

1. Reporte de asistencia
2. Reporte de estado de reservas

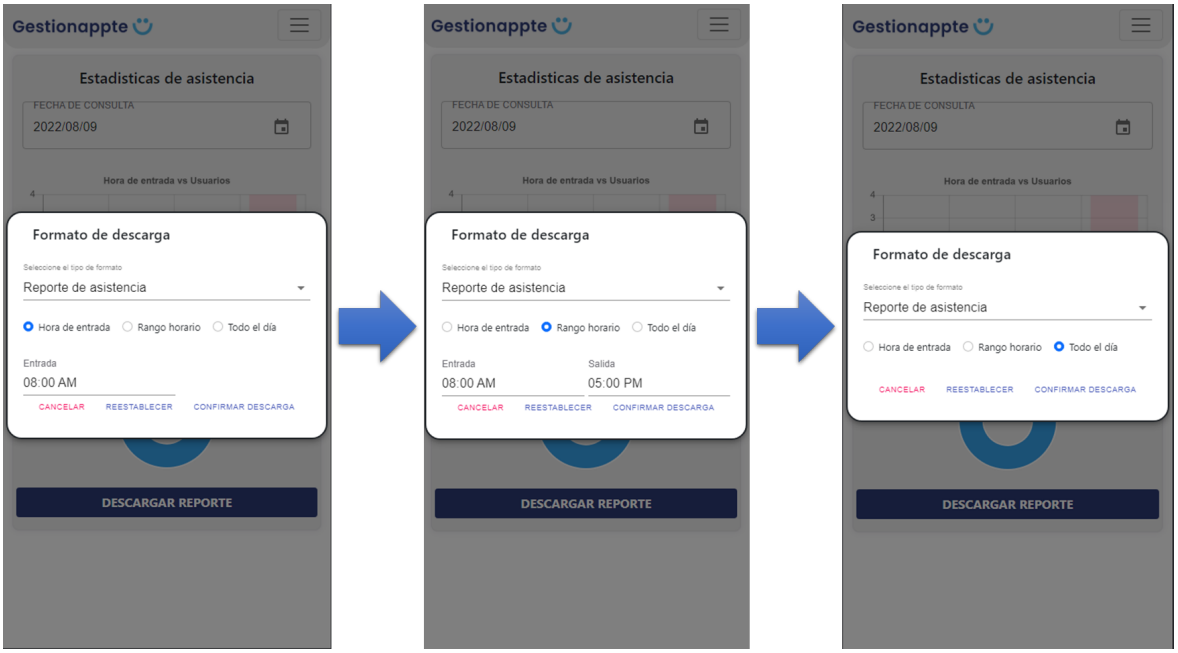
Cuando se elige, por ejemplo, reporte de asistencia, se habilitan unas opciones para el filtrado de la información:

1. Hora de entrada de los usuarios.
2. Rango horario de los usuarios (Hora de entrada y salida).
3. Todo el día (desde las 7:00 hasta las 19:00, horario laboral de Cidenet S.A.S).

Tabla 14. Pasos secuenciales para seleccionar el tipo de reporte

#	Pasos para la elección del tipo de reporte
1	Desde la vista de las estadisticas de asistencia, hacer click en "DESCARGAR REPORTE"
2	Una vez desplegada la modal, seleccionar el tipo de reporte

Figura 50. Tipos de filtro de la modal de estadísticas



Cuando el usuario-administrador elige una opción de filtrado, se confirma la descarga, de forma que se verifica que se hayan cargado datos de la base de datos y el documento excel se haya creado. Después, se despliega una modal con detalles generales del documento y se procede a descargar el reporte.

Tabla 15. Pasos secuenciales para seleccionar el tipo de filtro en el reporte

#	Pasos para los filtros del tipo de reporte
1	Cuando se selecciona el tipo de reporte, dar click en el tipo de filtrado
2	En caso de que el filtrado necesite algun tipo de tiempo, seleccionar el de preferencia
3	Click en "CONFIRMAR DESCARGA"

Figura 51. Descarga del reporte de asistencia

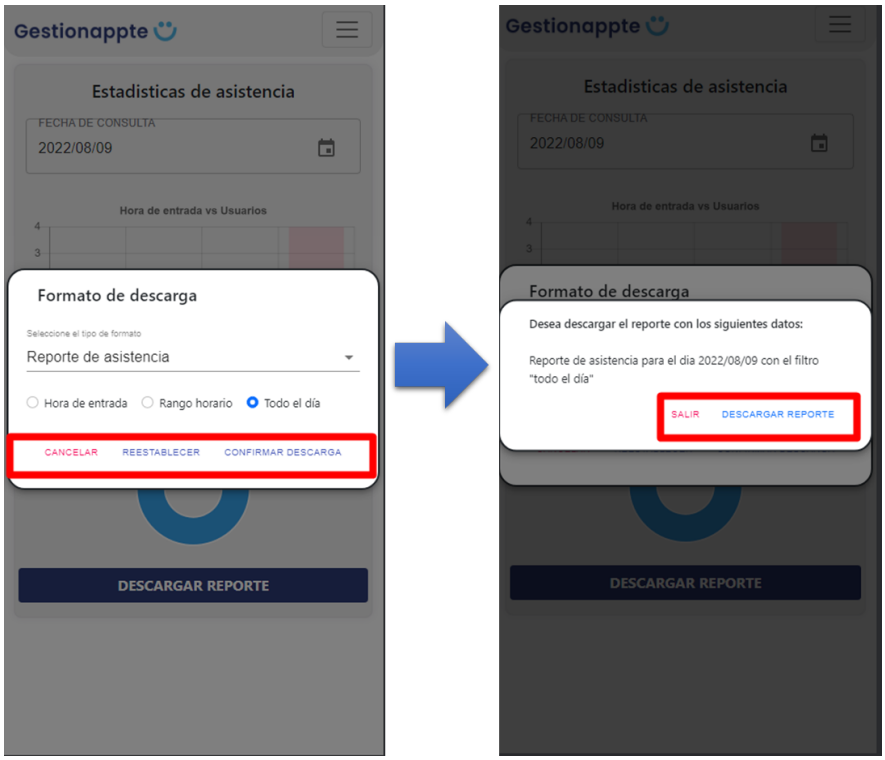
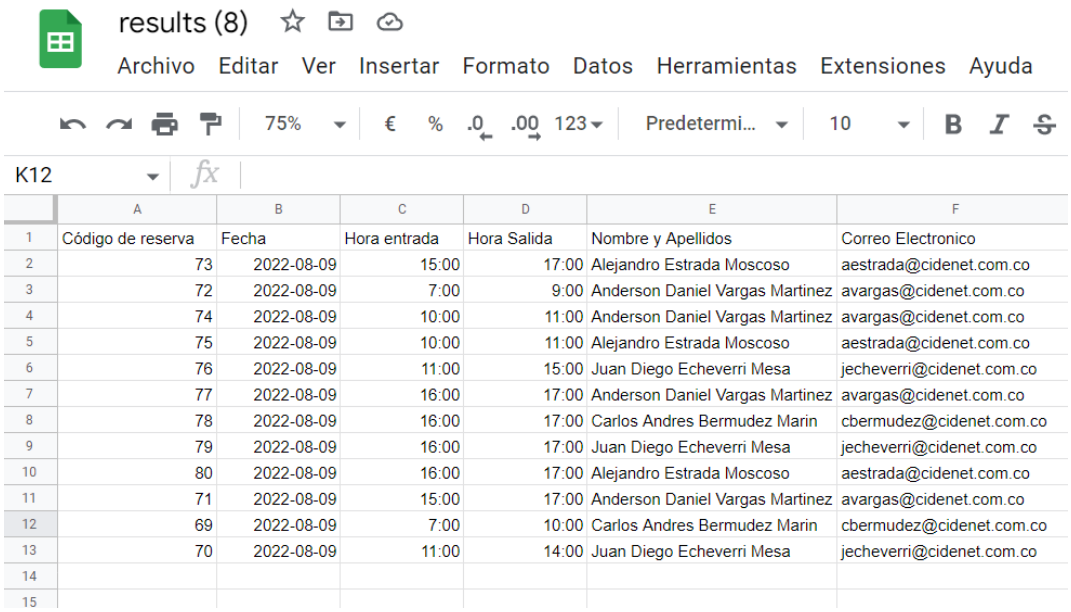


Figura 52. Formato del reporte descargado


results (8)

Archivo Editar Ver Insertar Formato Datos Herramientas Extensiones Ayuda

75% € % .0 .00 123 Predetermi... 10

K12

	A	B	C	D	E	F
1	Código de reserva	Fecha	Hora entrada	Hora Salida	Nombre y Apellidos	Correo Electronico
2	73	2022-08-09	15:00	17:00	Alejandro Estrada Moscoso	aestrada@cidenet.com.co
3	72	2022-08-09	7:00	9:00	Anderson Daniel Vargas Martinez	avargas@cidenet.com.co
4	74	2022-08-09	10:00	11:00	Anderson Daniel Vargas Martinez	avargas@cidenet.com.co
5	75	2022-08-09	10:00	11:00	Alejandro Estrada Moscoso	aestrada@cidenet.com.co
6	76	2022-08-09	11:00	15:00	Juan Diego Echeverri Mesa	jecheverri@cidenet.com.co
7	77	2022-08-09	16:00	17:00	Anderson Daniel Vargas Martinez	avargas@cidenet.com.co
8	78	2022-08-09	16:00	17:00	Carlos Andres Bermudez Marin	cbermudez@cidenet.com.co
9	79	2022-08-09	16:00	17:00	Juan Diego Echeverri Mesa	jecheverri@cidenet.com.co
10	80	2022-08-09	16:00	17:00	Alejandro Estrada Moscoso	aestrada@cidenet.com.co
11	71	2022-08-09	15:00	17:00	Anderson Daniel Vargas Martinez	avargas@cidenet.com.co
12	69	2022-08-09	7:00	10:00	Carlos Andres Bermudez Marin	cbermudez@cidenet.com.co
13	70	2022-08-09	11:00	14:00	Juan Diego Echeverri Mesa	jecheverri@cidenet.com.co
14						
15						

En el reporte descargado se cuenta con información diciente sobre las reservas de los usuarios, como:

1. Código de la reserva
2. Fecha de la asistencia
3. Hora de entrada y salida
4. Nombre y Apellidos del usuario
5. Correo electrónico de contacto

Finalmente, es importante mencionar que el segundo tipo de reporte, que es sobre el estado de las reservas, contiene exactamente los mismos atributos que el de la reserva con la adición del estado de la reserva.

6.3.6 Envío de notificaciones

Como se ha mencionado en anteriores páginas, el envío de notificaciones y la lógica interna con Gestionappte, se activa una vez el usuario reserva un espacio de trabajo. Las acciones que se ejecutan en esta sección son las siguientes:

1. Envío de recordatorio por email 24 horas antes de la reserva.
2. Envío de recordatorio por email 1 hora antes de la reserva.
3. Reservas que no sean confirmadas en la hora previa a la reserva y que tengan con respecto al horario actual una diferencia de menos de 10 minutos, serán canceladas de manera automática , haciendo que el espacio quede disponible para una futura reserva.

Figura 61. Ejemplo de una reserva y los correos recibidos en el buzón de gmail

Figura 53. Recordatorio de reserva 24 horas antes

Figura 54. Recordatorio de confirmación o cancelación de reserva 1 hora antes

En la figura 54, se visualiza el recordatorio de confirmación o cancelación que se encarga de manera directa de realizar estas dos acciones sin la necesidad de ir hasta la plataforma. Se ejecuta una petición al backend que confirma o cancela la reserva.

6.3.7 Nivel de satisfacción del grupo piloto de colaboradores

En este último apartado de resultados se realizaron dos encuestas, la primera previa al proyecto y la segunda una vez el MVP de Gestionappte fue presentado al grupo de colaboradores que probaron la aplicación. Esto con el fin de hacer una comparativa sobre el método mediante tickets a la mesa de ayuda y la solución que ofrece Gestionappte.

Experiencia de los colaboradores con la solución actual:

Figura 55. Encuesta 1

Figura 56. Encuesta 2

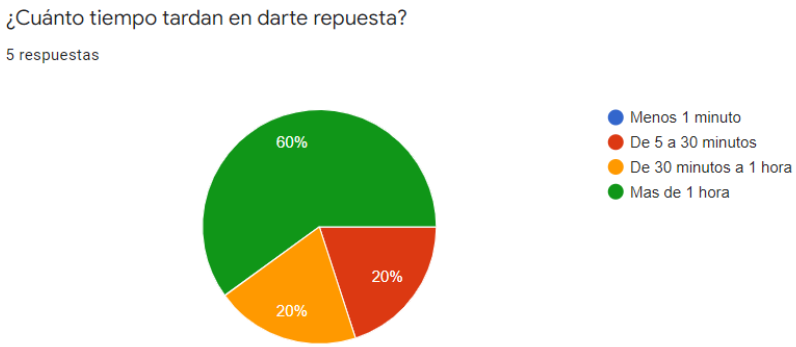
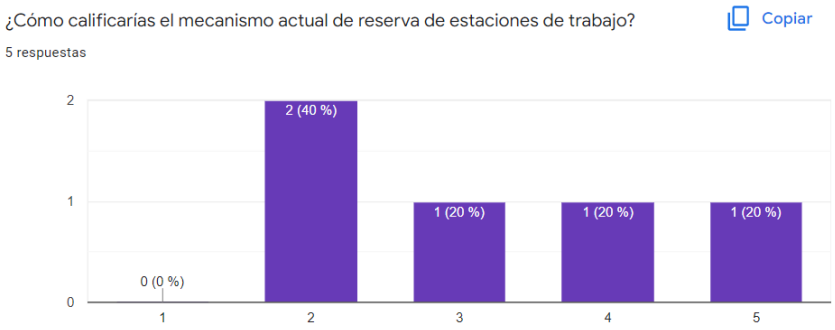


Figura 57. Encuesta 3



Una vez se presentó la solución al grupo piloto se envió la misma encuesta, con el motivo de conocer su experiencia, analizar mejoras y determinar cambios con respecto a los resultados de la primera encuesta.

Experiencia de los colaboradores con la solución de Gestionappte:

Figura 58. Encuesta 4

Figura 59. Encuesta 5

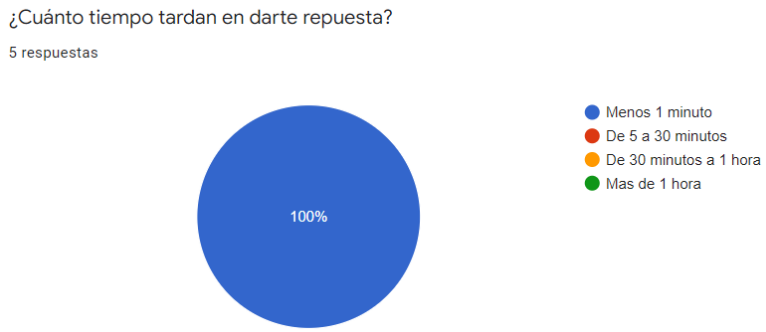
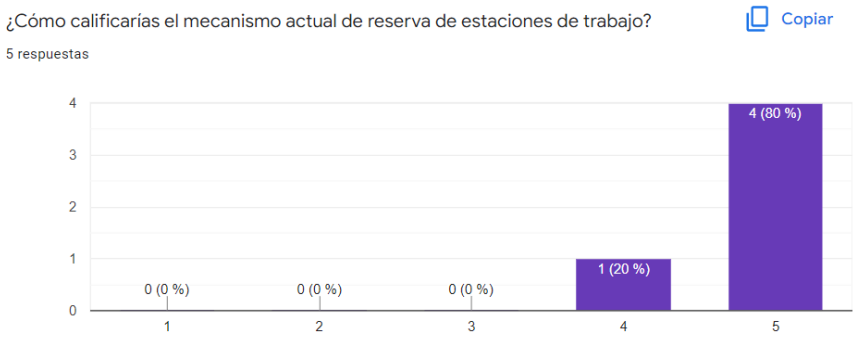


Figura 60. Encuesta 6



7. Conclusiones

- Al realizar el proyecto de Gestionappte se presenciaron diferentes etapas que existen cuando se requiere desarrollar software de calidad dentro de una empresa.
- El diseño responsive que se realizó en el frontend debe ser probado en diferentes resoluciones de pantalla, es posible que configuraciones que el desarrollador establezca para un tipo de pantalla funcione correctamente, pero al cambiar el dispositivo, esos mismos cambios puedan empeorar la experiencia del usuario dentro de la aplicación.
- En los momentos de análisis de las historias de usuario, se plantean casi siempre los casos ideales de las funcionalidades a desarrollar. Esto lleva a que a menudo se estimen las actividades con pocos tiempos de holgura, lo que provoca que los cumplimientos del sprint se vean afectados.
- La arquitectura de microservicios utilizada para estructurar la integración entre los servicios de la aplicación, se desarrolló de manera satisfactoria, dando la posibilidad de aumentar la cantidad de microservicios y el futuro alcance que Gestionappte pueda tener.
- Existen múltiples paquetes o librerías en los frameworks de desarrollo actuales que simplifican la experiencia del desarrollador en el momento de implementar alguna funcionalidad dentro de la aplicación.
- Un buen trabajo en equipo es la clave del éxito, tener la comunicación necesaria con los líderes técnicos, diseñadores e incluso otros desarrolladores es necesario para comparar enfoques, solucionar errores o realizar recomendaciones que prevengan futuros bugs dentro de la aplicación y la experiencia de usuario sea la más cercana a la planteada inicialmente.
- En la etapa de desarrollo es muy común realizar cambios en el alcance de las historias de usuario, esto debido a que en las etapas previas no se tienen en cuenta ciertos factores que en el momento de la codificación son evidentes.
- Los servicios en la nube como los de AWS o Google son soluciones que se están implementando de manera exponencial en la mayoría de empresas de software, pues reducen los gastos de hardware físico, costos energéticos y de seguridad.

- La integración de servicios en la nube debe ser previamente estudiada por el equipo de trabajo pues a menudo, este paso previo se evita y es cuando estas soluciones tienen valores monetarios que pueden ser incluso mayor que tener esos servicios físicos dentro de la empresa.

8. Anexos

A continuación se anexan unas imágenes que muestran las diferentes vistas de la aplicación desde una resolución web utilizando el navegador google chrome.

8.1 Login

Figura 61. Login Gestionappte

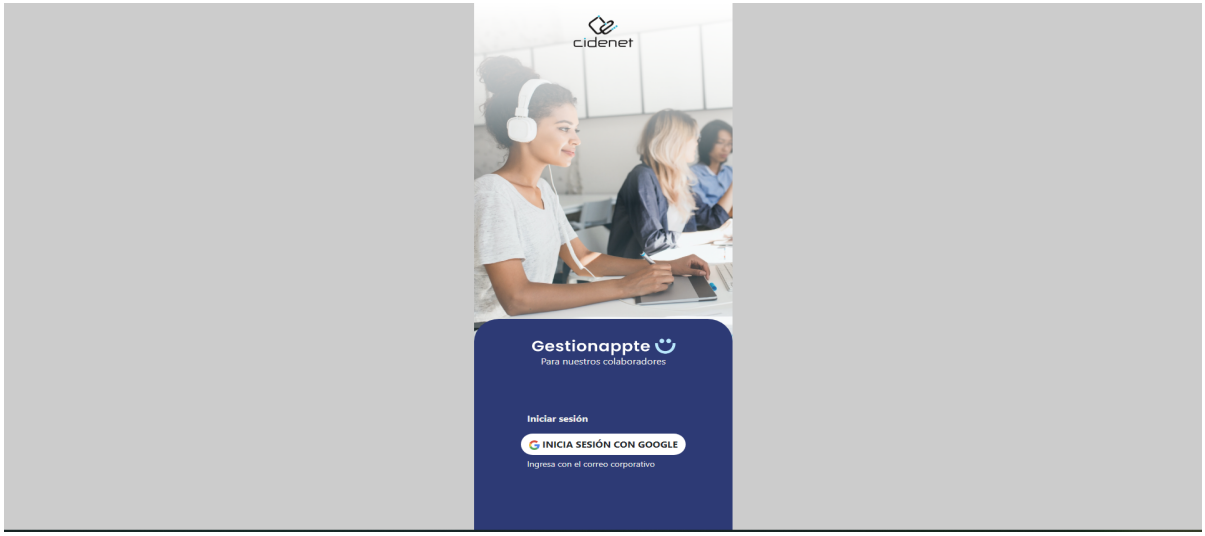


Figura 62. Pantalla intermedia de correo electrónico por google

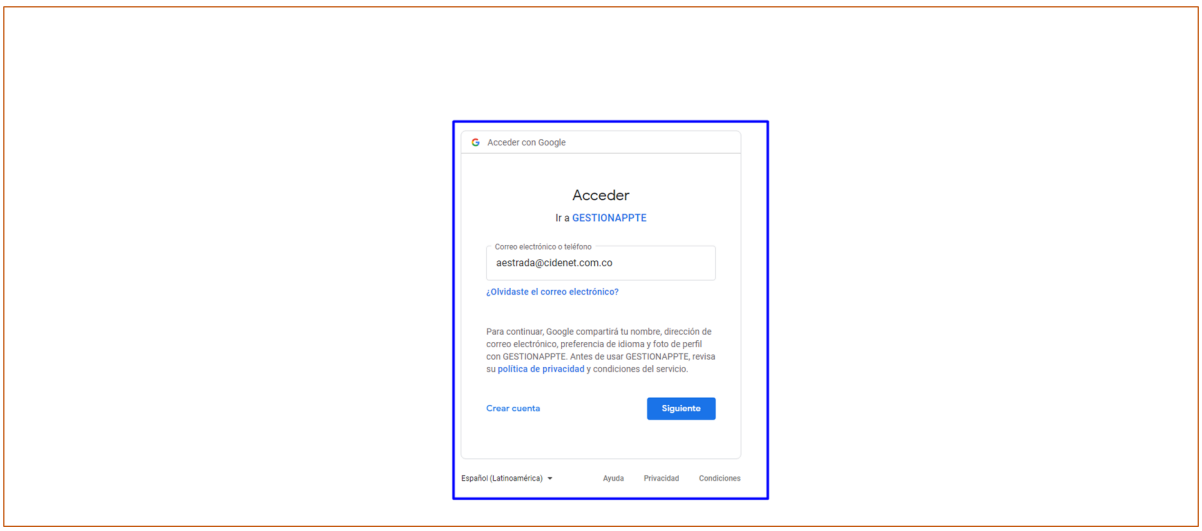
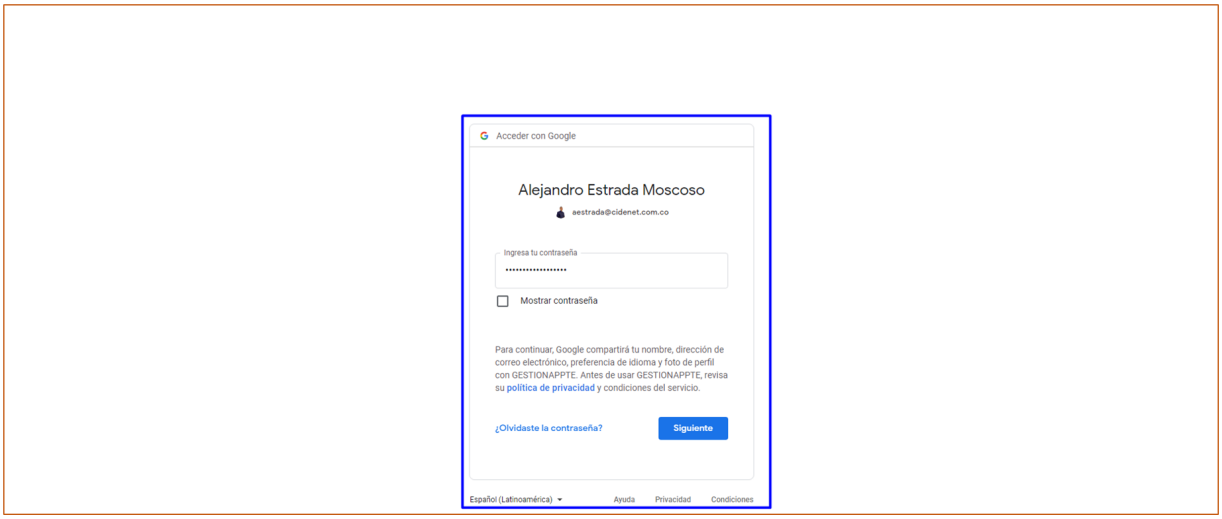


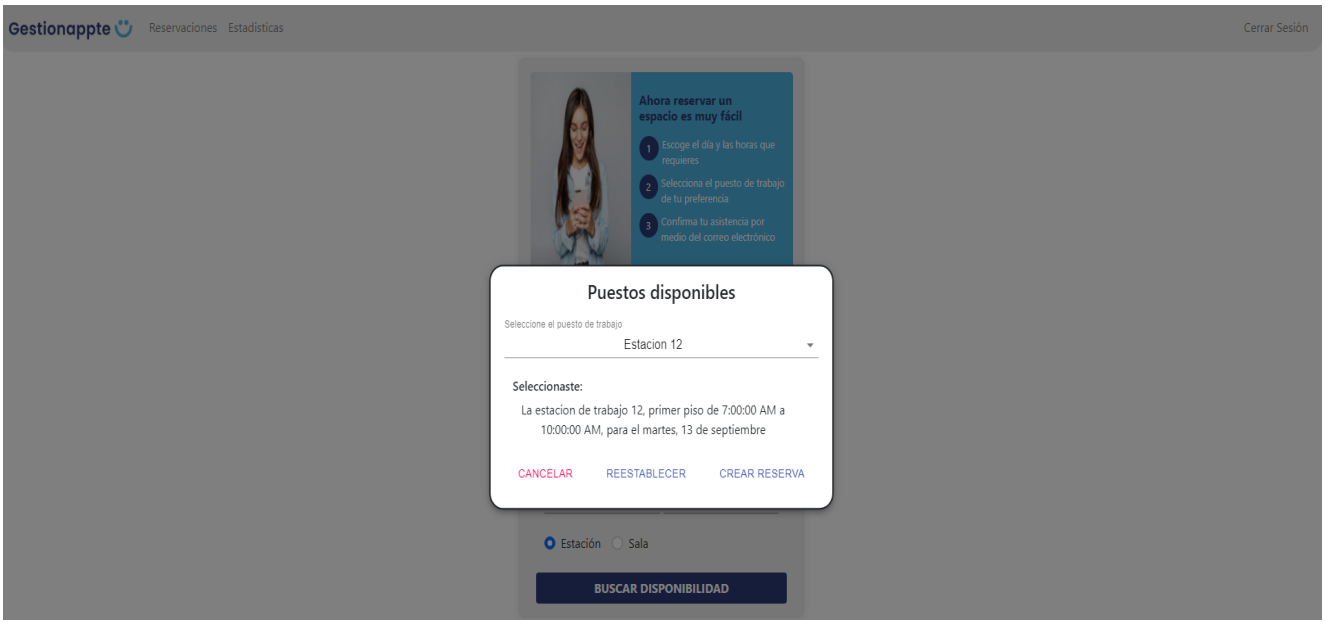
Figura 63. Pantalla intermedia de contraseña por google



8.2 Home

Figura 64. Vista principal del Home

Figura 65. Reserva de puesto de trabajo en Home



8.3 Reservaciones

Figura 66. Vista del listado de reservaciones

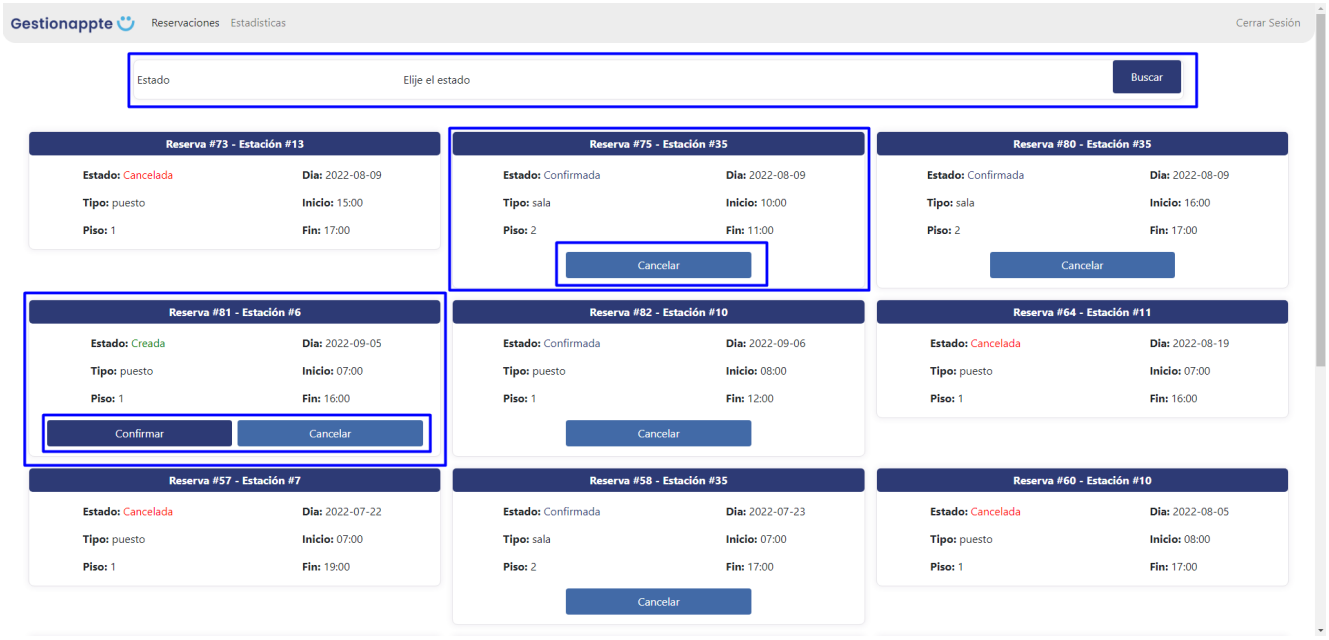
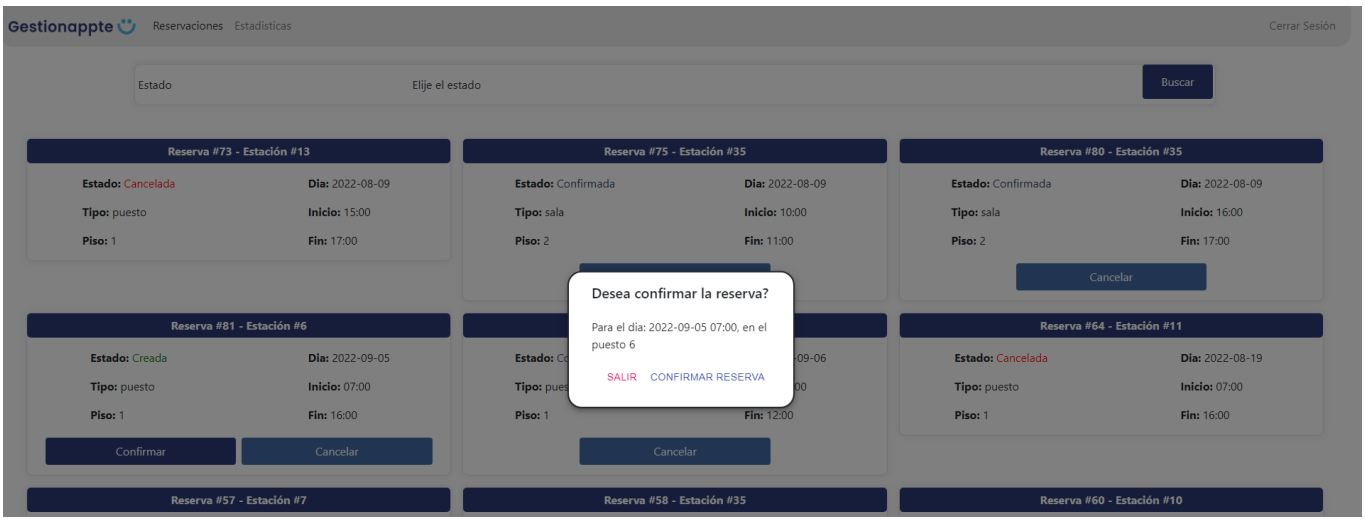


Figura 67. Modal de confirmación/cancelación de reservas



8.3 Estadísticas

Figura 68. Estadísticas de asistencia

Figura 69. Modal del formato de descarga

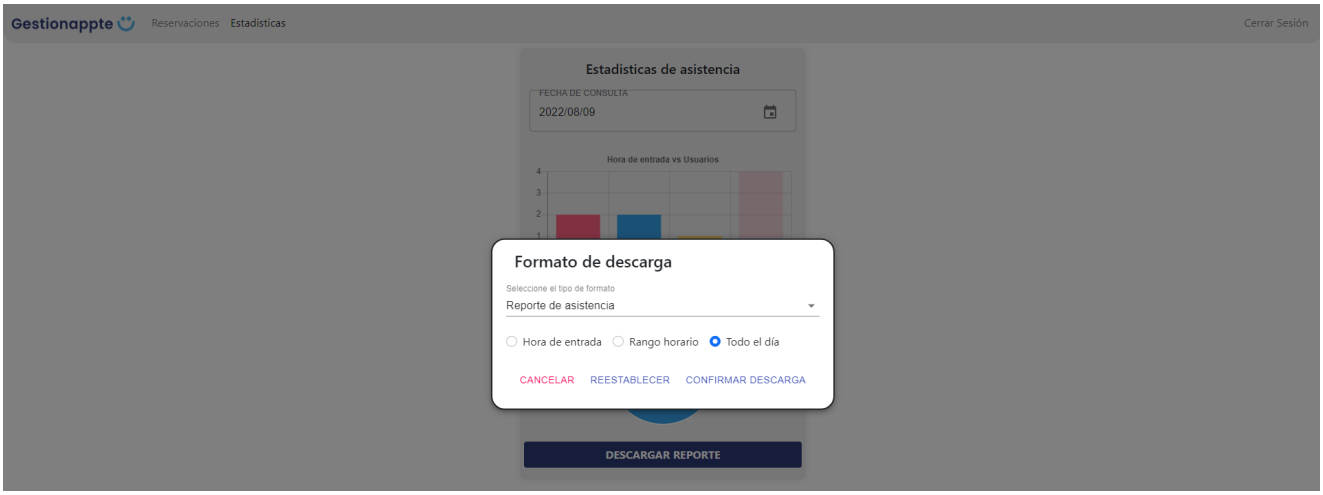
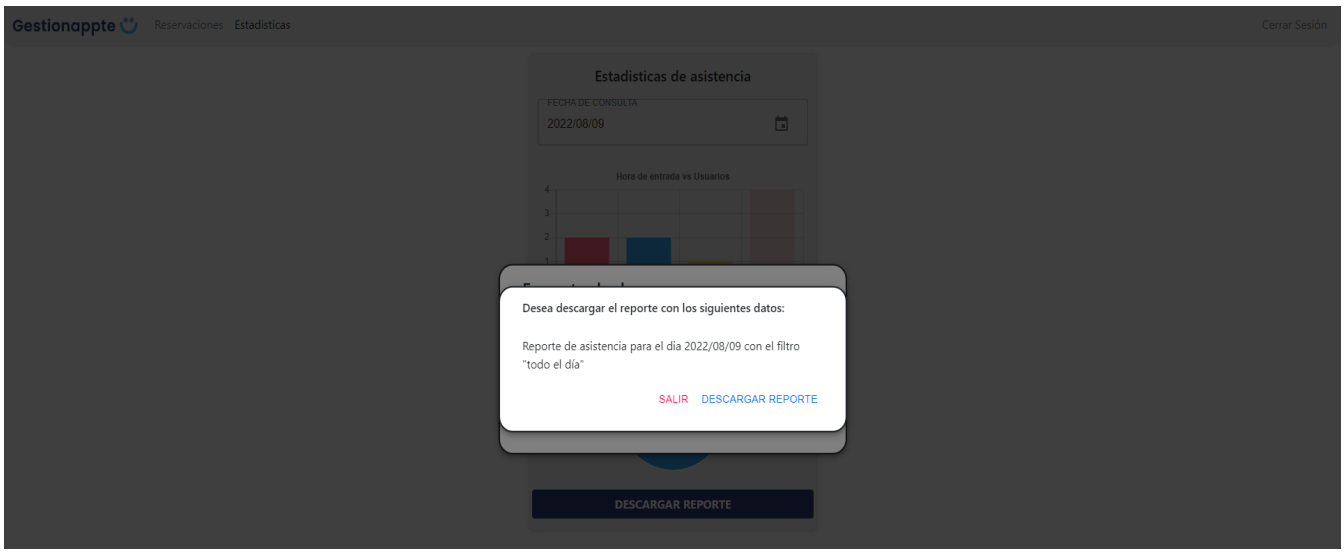


Figura 70. Modal de descarga de reporte



9. Referencias Bibliográficas

[1] D.Lopez, E.Maya «Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web» Julio, 2017. [En línea], Disponible en: https://documentos.redclara.net/bitstream/10786/1277/1/93%20Arquitectura%20de%20Software%20basada%20en%20Microservicios%20para%20Desarrollo%20de%20Aplicaciones%20Web.pdf

[2] R.D.Navarro & R.D.Cabrera «APLICACIÓN BASADA EN ARQUITECTURA DE MICROSERVICIOS» (2019). [En línea], Disponible en: https://eprints.ucm.es/id/eprint/62080/1/DIEGO_NAVARRO_APLICACION_BASADA_EN_ARQUITECTURA_DE_MICROSERVICIOS_4398577_313383662%20(1).pdf

[3] M.Moreno «análisis comparativo de herramientas orientadas a componentes web validado con un caso de estudio.», Agosto 2017. [En línea]. Disponible en: http://repositorio.espe.edu.ec/bitstream/21000/13539/1/T-ESPE-057416.pdf

[4] React Js, «React Js», 2022. [En línea]. Disponible en: https://es.reactjs.org/

[5]Spring Boot,«Spring Boot», 2022. [En línea]. Disponible en: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#documentation

[6]P.Tito «desarrollo de un sistema de gestión y control de procesos para la microempresa bazar y papelería san antonio utilizando spring boot.», Mayo 2020. [En

línea].Disponible:http://repositorio.utn.edu.ec/bitstream/123456789/10366/2/04%20ISC%20552%20TRABAJO%20GRADO.pdf

[7] Amazon, «Información general sobre Amazon Web Services», Agosto 2021. [En línea].Disponible:https://docs.aws.amazon.com/es_es/whitepapers/latest/aws-overview/aws-overview.pdf

[8] D. Hardt, Ed «The OAuth 2.0 Authorization Framework», Octubre 2012. [En línea]. Disponible: https://www.rfc-editor.org/rfc/rfc6749#section-1

[9] M.A. Duran, «Aprendiendo Git», Abril 2022. [En línea]. Disponible: http://leanpub.com/aprendiendo-git

[10] Stackoverflow «Aprendizaje Gitlab», 2022. [En línea]. Disponible: https://riptutorial.com/Download/gitlab-es.pdf

[11] S. Ravoof «GitLab vs GitHub: Descubre Sus Principales Diferencias y Similitudes». [En línea]. Disponible: https://kinsta.com/es/blog/gitlab-vs-github

[12] E.M. Suarez «¿Que es una base de datos relacional?». [En línea]. Disponible: http://www.uprh.edu/adem/Base%20de%20datos%20relacional.pdf

[13] M. Marqués «Bases de datos», [En línea]. Disponible: https://bdigital.uvhm.edu.mx/wp-content/uploads/2020/05/Bases-de-Datos.pdf

[14] L. Valencia «Introducción a SQL». [En línea]. Disponible: https://www.cs.us.es/cursos/bd/temas/BD-Tema-5.pdf

[15] A. Aliaga & M. Agustin «PostgreSQL». [En línea]. Disponible: https://iessanvicente.com/colaboraciones/postgreSQL.pdf

[16] Revista Digital Universitaria: DISEÑO WEB ADAPTATIVO O RESPONSIVO. Ciudad de Mexico, Mexico: Universidad Nacional Autonoma de Mexico, 2013, vol.14, nro.1.ISSN: 1067-6079

[17] R. Penciarolli, «Diseño web responsive», 2018. [En línea]. Disponible: https://imgbiblio.vaneduc.edu.ar/fulltext/files/TC130197.pdf

[18] Universidad de Alicante, Dept de la Computación e IA, «Introducción a los Servicios Web

RESTful», 2013, [En línea]. Disponible: http://www.jtech.ua.es/j2ee/restringido/cw/sesion11-apuntes.pdf

[19] CA technologies, «Una guía para el diseño de API y REST», 2015 [En línea]. Disponible: https://www.idglat.com/afiliacion/whitepapers/453164_A%20Guide%20to%20REST%20and%20API%20Design%20eBook-LAS.pdf

[20]J.Peñas, L.Reyero, «JSON LANGUAGE ORIENTED PROCESSING», 2012 [En línea]. Disponible: https://eprints.ucm.es/id/eprint/16693/1/Memoria_JLOPfinal.pdf

[21]Universidad de Alicante, Curso librerias web, «JSON», 2008 [En línea]. Disponible: https://si.ua.es/es/documentacion/mootools/documentos/pdf/json.pdf

[22] JSON Org, «Introducing JSON», [En línea]. Disponible: https://www.json.org/json-en.html

[23]Digital Guide IONOS, «JSON Web Token (JWT)», 2020 , [En línea]. Disponible: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/json-web-token-jwt

[24]C. Álvarez, «Introducción a JSON Web Token y la seguridad», 2017, [En línea]. Disponible: https://www.arquitecturajava.com/introduccion-a-json-web-token/

[25]R. Garcia, W. Calderon, J.C.Álvarez, «Simple Mail Transfer Protocol», [En línea]. Disponible: https://docplayer.es/53561262-Smtp-simple-mail-transfer-protocol.html

[26]J. Casas, «DISEÑO Y DESARROLLO DE UNA APLICACIÓN CLIENTE-SERVIDOR USANDO COMO MÉTODO DE TRANSFERENCIA DE DATOS CLIENTES DE CORREO ELECTRÓNICO.», 2018, [En línea]. Disponible: https://repositorio.upct.es/xmlui/bitstream/handle/10317/749/pfc2769.pdf

[27]K.Schwaver, J.Sutherland «La Guía de Scrum», 2020. [En línea]. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

[28]Deloitte «Scrum: roles y responsabilidades», 2022. [En línea], Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html

[29]L. Lozada, «EL TELETRABAJO: UNA MODALIDAD DE TRABAJO EFICIENTE QUE SE IMPONE COMO TENDENCIA GLOBAL», 2016 [En línea]. Disponible en: https://repository.unimilitar.edu.co/bitstream/handle/10654/15855/LozadaElizaldeLuisCarlos2016.pdf

[30]A. González, G. Nieto, «El proceso de creación y evolución del Producto Mínimo Viable en las startups de software», 2019, [En línea]. Disponible en: https://dspace.ort.edu.uy/bitstream/handle/20.500.11968/4075/Material%20completo.pdf?sequence=-1&isAllowed=y#:~:text=Un%20MVP%20(M%C3%ADnimo%20Producto%20Viable,la%20menor%20cantidad%20de%20recursos