**Dirección de Disponibilidad de Predios**

**Anillo Vial Periférico**

**Requerimientos Técnicos**

**Versión 1.0**

**Historia de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 13/08/2020 | 1.0 | Primera versión | Carlos Rafael Q. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[1. Descripción de Escenario 4](#_Toc47915747)

[2. Tecnologías Recomendadas 4](#_Toc47915748)

[2.1. React 4](#_Toc47915749)

[2.1.1. Babel 5](#_Toc47915750)

[2.1.2. Webpack 5](#_Toc47915751)

[2.1.3. React router 6](#_Toc47915752)

[2.2. REST API 7](#_Toc47915753)

[2.3. Node.JS 8](#_Toc47915754)

[2.4. Express.JS 10](#_Toc47915755)

[2.5. Npm 10](#_Toc47915756)

[2.6. PostgreSQL 11](#_Toc47915757)

[2.6.1. Sequelize 12](#_Toc47915758)

**Requerimiento Técnico**

1. Descripción de Escenario

El objetivo principal de este proyecto es implementar una plataforma web a través de la cual se puedan analizar datos relacionados a las labores de gestión predial de La Dirección de Disponibilidad de Predios. Esta construcción se llevará a cabo considerando una solución moderna y simple.

La principal motivación para la selección de las tecnologías que a continuación se describen es que durante estos últimos 8 años los frameworks de Javascript están a la vanguardia del desarrollo web por su versatilidad, al poder ser usados tanto en la parte del cliente como en la del servidor, además tras el desarrollo de los frameworks más usados hay grandes empresas y usuarios que velan por su desarrollo de forma que actualmente la mayoría de navegadores web utilizados soportan su ejecución.

1. Tecnologías Recomendadas
   1. React

React es una biblioteca Javascript que permite crear interfaces de usuario interactivas basadas en componentes reutilizables.

Fue creada por Facebook en 2011, su creación se produjo como necesidad de resolver una serie de problemas que existían en la red social, entre ellos una mejora del rendimiento, ya que, hasta ese momento la cantidad de conexiones que se debían realizar entre las vistas y los datos era excesiva ralentizando el rendimiento de la red social.

Además del rendimiento esta herramienta facilitaba enormemente el desarrollo ágil de componentes de interfaces de usuario reutilizables, por lo que se ha convertido en uno de los frameworks open-source con más proyección de futuro hoy en día. En 2013, su código fue liberado, convirtiéndose en una de las librerías más utilizadas por empresas entre las que se encuentran Instagram, Airbnb, Netflix o Dropbox.

Los componentes de React implementan el método de renderizado, o render, que toma una serie de parámetros de entrada y genera la salida correspondiente en el navegador. Además de la propia lógica del componente, el resultado del render depende, fundamentalmente, de dos tipos de valores:

* Propiedades: Conocidas como props. Son aquellos valores que se proporcionan al componente en el momento de su creación y cuyo valor es utilizado para el posterior renderizado del mismo. Su valor es inmutable a lo largo de todo el ciclo de vida del componente.
* Estado: Conocido como state. Son aquellos valores que representan el estado interno de un componente en un momento determinado del ciclo de vida. Un cambio en el estado del componente provocará necesariamente un renderizado del mismo.

Dentro del proceso de carga de una página, la tarea más pesada es la de actualizar la parte visible del navegador. React utiliza un DOM virtual, sobre el que realiza las actualizaciones correspondientes a los renderizados, para posteriormente compararlo con el DOM original y realizar únicamente los cambios mínimos necesarios en la página, en lugar de recargarla por completo.

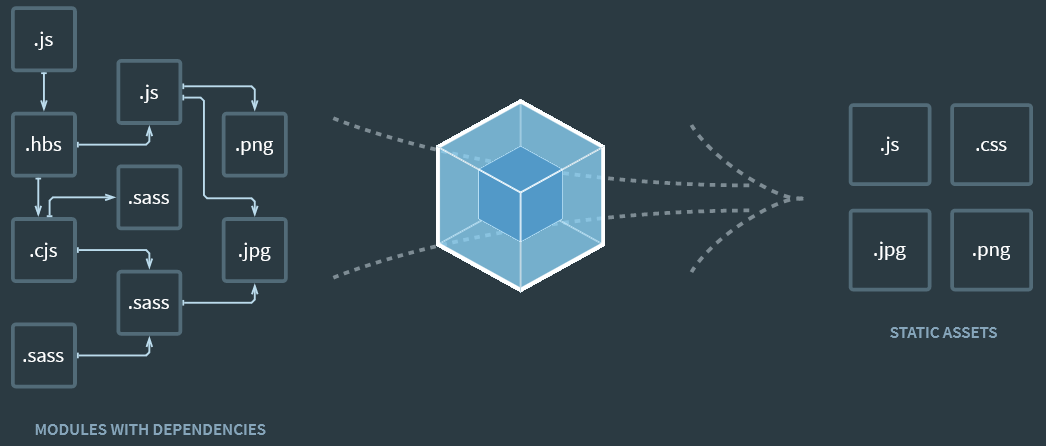
* + 1. Babel

React se desarrolla mediante una extensión de sintaxis similar a XML para ECMAScript. Además, al estar basado en ECMAScript es sencillo de utilizar si se conoce JavasScript ya que utiliza la mayoría de sus funciones y permite la inserción de código HTML facilitando la creación y legibilidad de los componentes. Sin embargo, no es una sintaxis destinada a ser interpretada por los navegadores web, sino que debe ser preprocesada a través de transpiladores que lo transforman en lenguajes que sean interpretables por el navegador.

En este contexto aparece Babel, un transpilador de código que transforma el JSX de React, en código Javascript. Además de ello, se encarga de transformar las funcionalidades de Javascript más modernas, cuya compatibilidad no es total, al estándar ECMAScript5, compatible con todos los navegadores actuales. Es posible que, con el paso de los años y las distintas actualizaciones de los navegadores, el uso de los transpiladores no sea necesario, pero, por el momento, no existe otra alternativa para poder utilizar React.

* + 1. Webpack

Webpack se define como un empaquetador de módulos o bundler, aunque podríamos definirlo como la herramienta de compilación utilizada por React para generar los archivos definitivos que se utilizarán en el servidor web. A grandes rasgos, es un task runner que se ocupa de procesar unos archivos de entrada y generar los correspondientes archivos de salida (como se puede observar en la figura).



Mediante un archivo JSON, se definen una serie de reglas que se aplicarán al código fuente para obtener el resultado definitivo. Entre ellas, las más comunes son:

* Transpilación del código Javascript ES6+ y JSX a Javascript ES5 utilizando Babel.
* Gestión de dependencias.
* Concatenación del código.
* Minimización y ofuscación del código.

Esta herramienta es muy útil cuando se requiere desarrollar aplicaciones web con una filosofía modular, es decir, distribuyendo el código en módulos que a su vez son dependencias de otros módulos.

* + 1. React router

React Router es una librería que permite la gestión dinámica de rutas para convertir aplicaciones React en Single Page Applications. Utilizando esta librería, se aprovecha el potencial de React ya que únicamente se realiza una petición HTTP al servidor desde el navegador la primera vez que necesitas cargar la página o bien cuando la actualices manualmente, y el código que se envía como respuesta es siempre el mismo independientemente de la ruta a la que se acceda. A partir de ese momento, todos los cambios en la aplicación o modificaciones en la ruta de la aplicación son gestionadas desde cliente, modificando únicamente aquellos componentes que varían de una ruta a otra.

Esta es una de las características que permiten un incremento muy notable en el rendimiento de las aplicaciones React puesto que se elimina por completo el tiempo de transferencia HTTP de los servidores tradicionales. Además, al delegar gran parte de la lógica al lado del cliente, la escalabilidad es mucho mayor.

Sin embargo, para la carga de datos desde la BBDD aún se necesita realizar peticiones a servidor para ello se utiliza una API mediante la cual el servidor retornará los datos necesarios.

* 1. REST API

REST cambió por completo la ingeniería de software a partir del 2000. Este enfoque de desarrollo de proyectos y servicios web fue definido por Roy Fielding, el padre de la especificación HTTP y uno los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación ‘Architectural Styles and the Design of Network-based Software Architectures’. En el campo de las APIs, REST (Representational State Transfer- Transferencia de Estado Representacional) es, a día de hoy, el estándar del desarrollo de servicios de aplicaciones.

REST le agrega una capa muy delgada de complejidad y abstracción a HTTP. Mientras que HTTP es transferencia de archivos, REST se basa en la transferencia de recursos.

Las API RESTful son aquellas que están diseñadas con los conceptos de REST:

* Recursos: toda la información que proporcione la API debe ser un recurso.
* URI: los recursos en REST siempre se manipulan a partir de la URI, identificadores universales de recursos.
* Acción: todas las peticiones a tu API RESTful deben estar asociadas a uno de los verbos de HTTP.

Dentro de una API RESTful todo recurso que sea expuesto debe tener una URI única como identificador, los clientes que quieran acceder a estos recursos deberán hacerlo con el método HTTP correspondiente:

* **GET** para obtener/leer un recurso.
* **POST** para escribir/crear un nuevo recurso.
* **PUT** para modificar un recurso, debe mandar toda la información completa del recurso, no se puede sobrescribir partes puntuales del recurso.
* **DELETE** para borrar un recurso.

Para una API RESTful se debe planear primero, pensar en los casos de uso por adelantado, diseñar los recursos y los recursos de manera correspondiente.

REST es muy útil cuando:

* Las interacciones son simples.
* Los recursos de tu hardware son limitados.

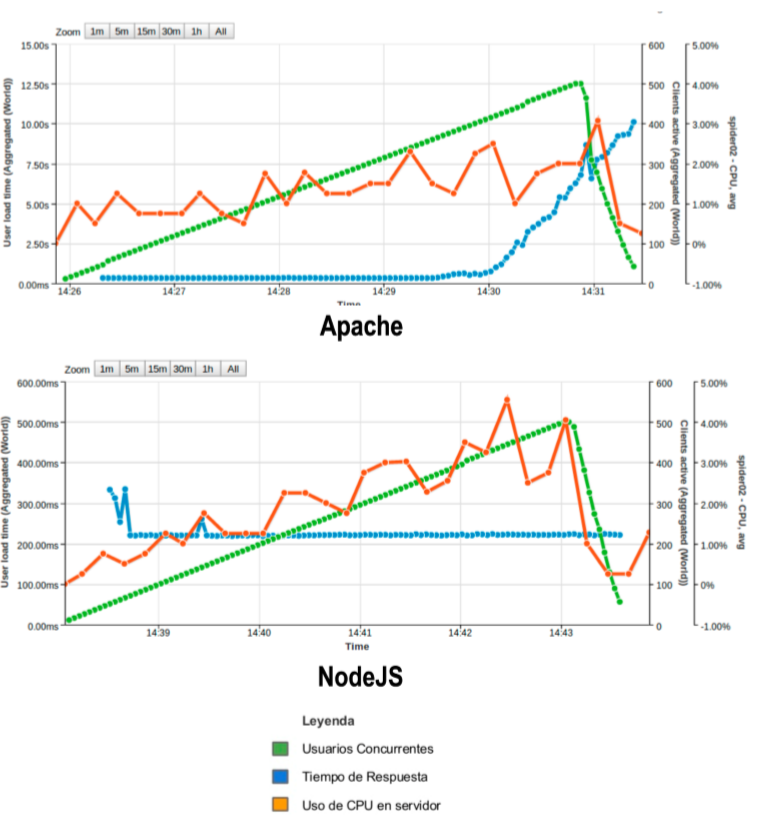
Además, REST nos brinda ventajas como:

* **Separación entre el cliente y el servidor**: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.
* **Visibilidad, fiabilidad y escalabilidad**: la separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.
* **La API REST siempre es independiente del tipo de plataformas o lenguajes**: la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.
  1. Node.JS

Node.JS es un entorno en tiempo de ejecución multiplataforma que suele usarse en la capa del servidor (pero no limitada a esta) y que está basado en ECMAScript. Fue creado con el objetivo principal de poder construir plataformas web con una alta escalabilidad y aquellos entornos tradicionales sufrían cuando se veían sometidos a una alta concurrencia de usuarios. Foundation [2009]

Al estar basado en ECMAScript el lenguaje en el que se codifica es casi idéntico a JavaScript ya que utiliza el motor de código V8, al igual que el navegador Google Chrome. Las principales ventajas de este entorno son las siguientes:

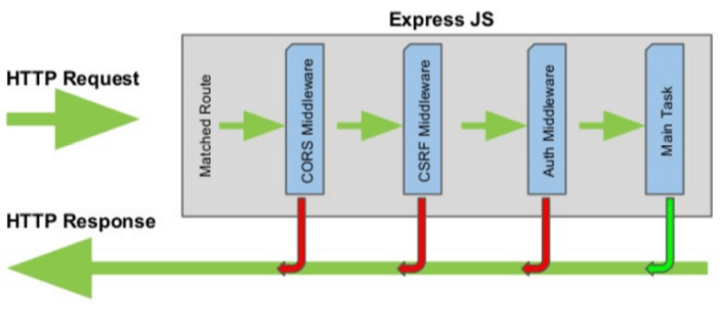
* Se trata de un entorno asíncrono, por lo que permite procesar una gran cantidad de solicitudes sin que su rendimiento se vea afectado. La forma en la que atiende las peticiones no se asemeja a los entornos tradicionales en los que para cada petición se asigna un hilo del SO, si no que cada conexión ejecuta un evento dentro del entorno que se traduce en un hilo del SO que se ejecuta en bucle, esto permite ahorrar mucho tiempo ya que por la propia naturaleza del motor, este no permite bloqueos y porque no se bloquea directamente cuando realizas una llamada de E/S.
* Orientado a eventos, la arquitectura está dirigida por eventos ya que por un lado es más útil que utilizar la orientada a objetos en este tipo de entornos. Esto es debido a que el funcionamiento del servidor se basa en solicitudes que requieren una respuesta, es decir, se realiza una conexión, se reciben ciertos datos y se devuelve la respuesta, estas acciones son mucho más cercanas a eventos que a un sistema puramente interactivo. Además, esta orientación permite que las operaciones no bloqueantes informen de que han finalizado mediante señales.
* Mayor tasa de transferencia, este entorno permite unas tasas de transferencia o throughput mucho mayores a los entornos tradicionales, la desventaja en este caso es que la latencia puede verse incrementada, por lo que depende de la orientación que tenga la plataforma web. Sin embargo, cuando la concurrencia crece, esta latencia suele tener una tendencia lineal como se puede observar en la figura.
* Modular, al igual que la mayoría de librerías y frameworks tiene una estructura modular que permite la carga y descarga de diferentes módulos que amplían la funcionalidad base y permiten unas gestión y actualización mucho más eficiente en comparación con otros entornos en los que la funcionalidad viene incluida en el core.



* 1. Express.JS

Express se trata de uno de los frameworks, si no el más utilizado de Node, y es la librería que permite el funcionamiento de otros frameworks ya que proporciona utilidades base para el desarrollo web, podría decirse que actúa como un programa que acepta las peticiones web y las maneja, entre estas utilidades se encuentran:

* Manejador de peticiones HTTP, que es el encargado de recibir y responder a las peticiones realizadas desde los clientes, acepta la gran mayoría de tipos de peticiones HTTP.
* Ajuste de aplicaciones web, permite realizar ajustes en la aplicación web como establecer el puerto de escucha, búsqueda de la ruta de las plantillas que se utilizan para responder a las peticiones.
* Actúa de middleware, esta es una de las utilidades más importantes ya que permite preprocesar las solicitudes a través de todos los middlewares necesarios ya sean propios o módulos de terceros. Esto es de utilidad para desestructurar las peticiones, clasificarlas en función del método utilizado, manejar los diferentes errores (tal y cómo se puede observar en la figura).



* 1. Npm

Npm es un gestor de paquetes y módulos que suele utilizarse junto con Node.JS. Consiste en un cliente de línea de comandos y un repositorio en línea de paquetes públicos y privados. Está completamente codificado en JavaScript. NPM [2013].

Las principales ventajas de utilizar este gestor son:

* Gestión eficiente y centralizada del código, permite buscar código empaquetado a través de su repositorio, al contrario que otros gestores en los que hay que buscar en varios repositorios.
* Gestión de dependencias, permite realizar las instalaciones y actualizaciones de forma ordenada, dando lugar a realizar despliegues del proyecto sin necesidad de mover todas las dependencias y con la ejecución de comandos sencillos.
* Soporte de la comunidad, al disponer de repositorios públicos y privados existe una amplia comunidad que se dedica al mantenimiento y revisión de los paquetes pudiendo elegir entre gran variedad.
  1. PostgreSQL

Las bases de datos relacionales son uno de los tipos más populares que existen, y su uso nos permite crear sitios web de toda clase. Entre los sistemas de bases de datos relacionales más utilizados se encuentra uno que es conocido como PostgreSQL, que dicho sea de paso está entre los más usados a nivel mundial.

PostgreSQL, o simplemente Postgres, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima.

Siendo uno de los sistemas de bases de datos más avanzados y usados del mundo, PostgreSQL ventajas como:

* **Es de código abierto**: una de las principales razones por la cual PostgreSQL se ha vuelto tan popular es que se trata de un sistema de código abierto. Esto ha permitido que una gran comunidad de desarrolladores crezca para respaldarlo y continuar mejorándolo. Gracias a todo el apoyo con el que cuenta ha logrado transformarse en uno de los mejores gestores de bases de datos a nivel mundial.
* **Es gratuito**: como cabe esperarse se trata de un sistema totalmente gratis, no tenemos que pagar nada por utilizarlo. Cualquier persona es libre de descargar PostgreSQL desde su sitio web oficial y darle uso sin ningún costo.
* **Es multiplataforma**: una característica genial que de hecho es común en muchos grandes proyectos de código abierto es el hecho de que se trata de software multiplataforma, es decir, es un software que puede correr bajo distintos entornos y sistemas operativos, y es compatible con muchos de los [servidores web](https://blog.infranetworking.com/servidor-web/) más populares como Apache, Nginx y LiteSpeed por mencionar algunos.
* **Es fácil de usar**: la facilidad de uso de PostgreSQL es sin dudas otra de las principales características de este sistema. Su administración se vuelve muy sencilla por medio de paneles con PgAdmin, que básicamente viene a ser un phpMyAdmin orientado para PostgreSQL. La posibilidad de realizar diversos procedimientos en forma sencilla hace que PgAdmin sea ampliamente utilizado, aunque también permite realizar tareas más complejas, así que tanto novatos como usuarios expertos hacen uso de él.
* **Puede manejar un gran volumen de datos**: una característica extremadamente importante de PostgreSQL es su gran capacidad para el manejo de grandes volúmenes de datos, algo en lo que otros sistemas como MySQL aún no hacen tan bien. Las bases de datos de gran tamaño pueden hacer pleno uso del MVCC de PostgreSQL, resultando en un gran rendimiento. MVCC es un método de control que nos permite realizar tareas de escritura y lectura simultáneamente.
* **Soporte total de ACID**: otro punto muy importante que no se debe dejar de lado es el cumplimiento de ACID. ¿Qué es ACID? Estas siglas en inglés refieren a: atomicity, consistency, isolation y durability, que si lo traducimos al español básicamente hablan de la atomicidad, consistencia, aislamiento y durabilidad de las transacciones que se realizan en una [base de datos](https://blog.infranetworking.com/base-de-datos/). ¿Y por qué es tan importante? Porque tener soporte completo de ACID da la seguridad de que, si se produce una falla durante una transacción, los datos no se perderán ni terminarán donde no deban.
* **PostGIS -la extensión para datos espaciales**: uno de los puntos más relevantes es la capacidad de almacenar y trabajar con datos de tipo geométrico. Para ello surge PostGIS: la extensión espacial para PostgreSQL, una herramienta impulsada por OsGEO.
  + 1. Sequelize

Sequelize es un ORM basado en promise para Node.js y io.js. Soporta PostgreSQL, MySQL, MariaDB, SQLite y dialectos MSSQL, y ofrece un sólido soporte transaccional. Lee replicación y más.

Una de sus mejores prestaciones es el manejo de promise. Es muy común, y una buena práctica utilizar Q promises en Node y JavaScript.

Algunas de sus características incluyen:

* **Querying**: En lo que respecta al querying básico, la API es por demás completa. Tiene varios métodos de recuperar la información desde la base de datos y varios operadores para utilizar en query como $and, $or, $gt, $between, $contain, etc. La manera de construir queries es muy similar a *mongo*. Además, Sequelize tiene la propiedad del *include*. Se puede usar esta propiedad para hacer queries entre cuadros de relación.

En conclusión, hay buen soporte de las operaciones CRUD.

* **Scoping**: Otra de las características útiles de Sequelize es scoping. Permite definir queries comúnmente utilizadas que pueden ser reutilizadas más tarde. Los scopes pueden incluir los mismos atributos como finders comunes: where, include, limit etc.
* **Relaciones / Asociaciones**: El framework cubre todas las relaciones y asociaciones.
* **Transacciones**: Realizar transacciones es simple. Puedes crear transacciones simultáneas/parciales también.
* **Migraciones**: Las migraciones son muy importantes en un modelo relacional. Sequelize ofrece una CLI donde se pueden crear y ejecutar migraciones. Su esqueleto consiste básicamente en dos métodos, up y down para convertir al nuevo estado y revertir los cambios, respectivamente. Además, existe la queryInterface para alterar/crear cuadros. Es común usarlos en los métodos de migración descriptos anteriormente.

Sequelize tiene una librería hermana (umzug) para manejar las tareas de migración de ejecución y logging de manera programada.