Prototipo de aplicación web para el diseño de pavimentos rígidos

Por:

Juan Fernando Otoya García

Director:

Juan Carlos Martínez Arias

Pontificia Universidad Javeriana Cali

Facultad de Ingeniería y Ciencias

Departamento de Electrónica y Ciencias de la Computación

Ingeniería de Sistema y Computación

Santiago de Cali, octubre 2024

RESUMEN

El diseño de pavimentos ha evolucionado desde métodos empíricos basados en observaciones de campo hasta métodos mecánicos apoyados en la teoría de la mecánica de sólidos. Inicialmente, los diseños eran prueba y error, pero con el tiempo se incorporaron modelos matemáticos robustos, facilitados por el uso de computadoras, desarrollados a partir de investigaciones en los Estados Unidos desde mediados del siglo XX. Aunque estos modelos inicialmente requerían ábacos y tablas, hoy en día los avances tecnológicos permiten el uso de software especializado, aunque es crucial alimentar estos programas con datos de alta calidad para obtener buenos resultados.

En el caso de Colombia y Latinoamérica, aunque existen programas de software accesibles como IMT-PAVE y BS-PCAA, presentan limitaciones en cuanto a la parametrización y el manejo de datos. Este documento entrega una descripción de un proceso de ingeniería de software para la creación de una aplicación web que diseñe pavimentos rígidos. Abarcando desde los antecedentes y la formulación de objetivos, hasta el análisis de requerimientos, diseño, implementación, pruebas, y recomendaciones finales.

ABSTRACT

Pavement design has evolved from empirical methods based on field observations to mechanical methods supported by solid mechanics theory. Initially, designs were trial and error, but over time robust mathematical models were incorporated, facilitated using computers, developed from research in the United States since the mid-twentieth century. Although these models initially required abacuses and tables, today technological advances allow the use of specialized software, although it is crucial to feed these programs with high-quality data to obtain good results.

In the case of Colombia and Latin America, although there is accessible software such as IMT-PAVE and BS-PCAA, they have limitations in terms of parameterization and data management. This document provides a description of a software engineering process for the creation of a web application that designs rigid pavements. It includes everything from the background and formulation of objectives to the analysis of requirements, design, implementation, testing and final recommendations.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA TABLAS

1. INTRODUCCIÓN

Por más de 100 años el diseño de pavimentos se ha realizado utilizando métodos empíricos y racionales. Los empíricos se fundamentan en la observación en campo mientras que los racionales o mecánicos se centran en la teoría de la mecánica de sólidos (AASHTOware, 2023). Anteriormente, los métodos empíricos se basaban en prueba-error, pero actualmente ambas metodologías están soportadas con modelos matemáticos robustos que requieren del uso de computadoras (Federal Highway Administratio, 2023). Estos modelos se empezaron a desarrollar a mediados del siglo XX en la investigación Nacional de Pavimentos de los Estados Unidos de Norteamérica y aún continúan perfeccionándose. Diversos métodos de diseño se desprendieron de dicha investigación y en los años 80 se empezaron a utilizar. Sin embargo, el uso del computador no era factible para las empresas de diseño y construcción, por lo cual los desarrolladores como FHWA (Federal Highway Administratio, 2023), AASHTO (AASHTOware, 2023), PCA (Packard & Tayabji, 1985), entre otros, elaboraron gran cantidad de ábacos, nomogramas y tablas con los cuales los ingenieros realizaban sus diseños.

Actualmente, la accesibilidad al computador permite la sistematización de dichos métodos y existen varios softwares cada uno con ventajas y desventajas (Tabla 1). Es importante aclarar que cualquier software debe ser alimentado con datos de alta calidad de lo contrario se pierde la bondad del método.

Entre las ventajas de algunos de ellos está el que o bien son descargables o en línea, algunos con interfaz amigable y sencilla, algunos con ayuda para el ingreso de datos. Por otra parte, el software mexicano IMT-PAVE y los colombianos BS-PCAA y PCA Cálculo son los softwares libres más amigables para la forma como se adquieren los datos en Colombia y Latinoamérica, pero el primero está parametrizado con el tránsito mexicano mientras que los otros dos no son autocontenidos ya que las bases de datos se encuentran en otros programas de computador y deben ser procesadas para alimentarlos. Entre las desventajas de todos se encuentra el que las variables de entrada deben ser fabricadas previamente por el diseñador lo que implica un gran esfuerzo y por ende puede llevar de nuevo a las simplificaciones indeseables que ocurrían en el Siglo XX cuando el diseño se realizaba manualmente (García Aladín, La importancia de los computadores en el diseño de pavimentos., 2024).

Este documento sigue el proceso de ingeniería de software que se llevó a cabo de principio a fin para lograr crear una aplicación web que permita diseñar pavimentos rígidos. Los capítulos uno a cinco contiene los antecedentes, el planteamiento de objetivos, el marco teórico y marco conceptual que soportan el proceso de ingeniería de software. En el capítulo 6 se realiza el análisis de requerimientos, el capítulo 7 el diseño de las diferentes partes de la aplicación, el capítulo 8 las decisiones de implementación y resultado de esta y el capítulo 9 las pruebas que se llevaron a cabo. Finalmente se presentan algunas conclusiones y recomendaciones en el capítulo 10.

2. DESCRIPCIÓN DEL PROBLEMA

2.1 Planteamiento del problema

Un pavimento rígido es un tipo de infraestructura vial compuesto principalmente de concreto hidráulico. Su diseño se enfoca en distribuir las cargas vehiculares de manera eficiente a través de la losa de concreto, lo que lo hace ideal para carreteras de alto tráfico y condiciones adversas. La durabilidad y resistencia de los pavimentos rígidos dependen de factores como la calidad del concreto, el espesor de las losas, y la correcta consideración de las cargas que deberá soportar.

El diseño de pavimentos rígidos implica determinar espesores adecuados y otros parámetros estructurales para garantizar que el pavimento soporte las cargas vehiculares esperadas a lo largo de su vida útil. Este proceso requiere un análisis detallado de las condiciones del tráfico, las propiedades del suelo y el clima, entre otros factores. Sin embargo, uno de los mayores desafíos en Colombia es la falta de datos precisos sobre el tránsito y las cargas específicas en diferentes regiones. Esto genera una dependencia de modelos y bases de datos extranjeros, que no siempre reflejan con precisión las condiciones locales.

La metodología de la Portland Cement Association (PCA) es un enfoque ampliamente utilizado para el diseño de pavimentos rígidos. Esta metodología proporciona guías detalladas para dimensionar pavimentos de concreto teniendo en cuenta factores clave como las cargas, el tránsito proyectado y las propiedades del suelo. Aunque es un método robusto, su aplicación depende en gran medida de la calidad de los datos de entrada, especialmente los relacionados con los espectros de carga y el tránsito vehicular.

En Colombia, aunque se ha percibido una falta de datos locales, los pesajes de camiones disponibles en el programa "Tránsito" (alojado en https://www.topo3.com/) muestran que los datos sí existen, pero están dispersos y no se han procesado adecuadamente para fines específicos de diseño de pavimentos. En esta investigación, en colaboración con dos estudiantes de un proyecto dirigido de la profesora María Fernanda García, se agruparon los espectros de carga para cada departamento del país, lo que permite una mejor representación de las cargas reales a las que estarán sometidos los pavimentos.

El desarrollo de un software basado en la metodología PCA y alimentado con datos locales procesados permitirá una mejor adaptación del diseño de pavimentos rígidos a las condiciones colombianas. Este software cubrirá las deficiencias de las herramientas actuales, que utilizan datos no representativos de nuestras condiciones locales, y proporcionará una solución integral y accesible para ingenieros y diseñadores viales en el país.

Una problemática relevante es la falta de software especializados y parametrizados para las condiciones específicas de tráfico y materiales locales capaces de cumplir con las metodologías de diseño (INVIAS, 2018), (INVIAS, 2008). Una de las grandes limitaciones se presenta en la base de datos del tránsito, ya que Colombia cuenta con operativos en carretera que, aunque están disponible requieren alto procesamiento para alimentar el diseño de pavimento. Colombia cuenta con 206 700 km de carreteras (Leal Acosta, 2019) de las cuales únicamente a 9 343 km (4.5%) son administrados por la Nación (INVIAS, 2023) y solo algunas cuentan con básculas camioneras para el control de cargas. Puede verse que más del 95% de las vías no tiene como controlar los pesos vehiculares lo que sugiere que es muy importante sistematizar los operativos de pesajes disponibles directamente en el software de diseño y crear la posibilidad de actualizarla o ampliarla. La mayoría de los programas disponibles en el mercado están diseñados y parametrizados para regiones como América del Norte y México, lo que dificulta su aplicabilidad y precisión en otros contextos geográficos. Esta falta de adaptación a las condiciones locales limita la eficacia y la precisión de los resultados obtenidos.

En la Tabla 1 se muestra un análisis de los siguientes software: WinPAS 12 (ACPA, 2014), StreetPAVE 12 (ACPA, 2014), MnPAVE Rigid 3.002 (MnDOT, 2019), PerRoad (Timm, 2017), PavExpress (APA, 2022), AI SW-1 Thickness Design (Pavementdesigner.org, 2018), AASHTOWare Pavement ME Design (The designer's corner, 2023), IMT-PAVE (Instituto Mexicano de Transporte (IMT), 2022), PCAcálculo (Casto & Orobio, 2015) y BS-PCAA (Solano & Benavidez, 2022).

A pesar de la importancia de tener en cuenta las condiciones locales en el diseño de pavimentos, actualmente no existen herramientas de software adaptadas específicamente al contexto colombiano. La mayoría de los programas disponibles en el mercado están parametrizados para otras regiones, lo que limita su aplicabilidad y precisión en el país. Esta falta de herramientas especializadas dificulta la implementación de métodos de diseño más precisos y racionales.

La degradación prematura de las vías representa un problema significativo en Colombia, ya que afecta la eficiencia del transporte, aumenta los costos de mantenimiento y pone en riesgo la seguridad de los usuarios. La falta de herramientas de diseño adecuadas y la dependencia de métodos empíricos pueden contribuir a esta problemática al no tener en cuenta adecuadamente las condiciones específicas de cada proyecto.

En conclusión, el diseño de pavimentos en Colombia enfrenta desafíos relacionados con la precisión de los métodos de diseño, la adaptación de las normativas a las condiciones locales y la degradación prematura de las vías. Abordar estos desafíos requerirá el desarrollo de herramientas de software especializadas que permitan a los ingenieros civiles realizar análisis precisos y eficientes, teniendo en cuenta las características únicas del contexto colombiano.

Tabla 1. Perfil de las herramientas tecnológicas actuales

Número	Nombre	Empresa	Origen	En linea	Método de diseño	Fecha	Ventajas	Deventajas	Observaciones	Link
1	WinPAS 12	ACPA	USA	Descargable, Gratis 30 días. Costo US$300 con derecho a actualizaciones	AASHTO 93	Feb 27 de 2014	Soportado por la FHWA, cuenta con modelos matemáticos robustos, bases de datos de alta calidad, la interfaz es amigable.	Bases de datos norteamericanas, requiere 96 variables tomadas en campo para su adecuada parametrización y uso en un ambiente local.	Tiene nueva GUI (graphical user interface), e incluye pantallas de ayuda lo cual hace que el software sea facil de utilizar. Bases de datos de Norteamérica.	https://www.acpa.org/winpas/
1.1		ACPA	USA	Gratis, en línea	AASHTO 93				Calculadora de ESAL	http://apps.acpa.org/applibrary/ESAL/
2	StreetPave 12	ACPA	USA	Descargable, Gratis 30 días. Costo US$300 con derecho a actualizaciones.	PCA	Ago 17 de 2014	Soportado por la FHWA, cuenta con modelos matemáticos robustos, bases de datos de alta calidad, la interfaz es amigable.	Bases de datos norteamericanas, requiere 96 variables tomadas en campo para su adecuada parametrización y uso en un ambiente local.	Para pavimentos de concreto y subrasantes existentes, StreetPave 12 puede ser utilizado para estimar la vida util o criterios de falla. Bases de datos de Norteamérica.	https://www.acpa.org/streetpave/
2.1		ACPA	USA	Gratis, en línea	PCA				Calculadora de K apoyo	http://apps.acpa.org/applibrary/KValue/
3	MnPAVE Rigid 3.002	MnDOT	USA, Minnesota	Gratis, descargable	Mn Department of Transportation	May 13 de 2019		No es amigable, bases de datos de Minesota.	Todas las variables están para el estado de Minesota	https://www.dot.state.mn.us/app/mnpave/
4	PerRoad	Aubum University. NCAT	USA, Alabama	Gratis, descargable	Metodología Racional	Sep de 2017	Amigable y sencillo	Bases de datos de Alabama.	Permite diseñar por la metodología racional, semejante a la PCA, pero sus bases de datos son Norteamericanas.	https://www.cesdb.com/perroad.html
5	PavEXpress	SAPA y NAPA	USA, alabama	Gratis, en línea	AASHTO 93	Jun 9 de 2022	Amigable y sencillo	Bases de datos de Alabama.	Analiza todas las variables de esta metodología, muy completo. Dases de datos de Alabama. Es amigable.	https://pavexpress.com/session-2-new-pavement-designs-with-1993-and-1998-aashto/
6	Al SW-1 Thickness Design	Pavement Designer.org	USA	Gratis, en línea	PCA	2018	Amigable y sencillo	Bases de datos de Norteamérica	Analiza todas las variables de esta metodología, amigable. Bases de datos de Norteamérica.	https://www.pavementdesigner.org/projectTypes/street/concrete/projectLevelForm
7	AASHTOWare Pavement ME Design	AASHTO	USA	En línea, se requiere comprar licencia.	MEPDG	2023	Soportado por la FHWA, cuenta con modelos matemáticos robustos, bases de datos de alta calidad, la interfaz es amigable.	Bases de datos norteamericanas, requiere 96 variables tomadas en campo para su adecuada parametrización y uso en un ambiente local.	Software oficial de AASHTO. No se puede aplicar en Colombia por la complejidad de los estudios de campo que se requieren. Es amigable.	https://youtu.be/PsJiNc_HWE8?feature=shared
8	IMT-PAVE 3.2	IMT	México	Gratis, descargable	PCA	Jun 19 de 2022	Soportado por el Instituto Mexicano del Transporte, amigable, sencillo, ofrece ayudas para el cáculo de módulos de los materiales.	Bases de datos mexicanas.	Bases de datos méxicanas	https://imtpave.com/
9	BS-PCA	UNICAUCA	Colombia	Gratis, descargable	PCA	2003		No ofrece ayudas de cálculo, no tiene bases de datos.	Sin bases de datos	https://drive.google.com/file/d/1Z5JMnZv_ZkOv0VexeDUq0HlT-UURi80Z/view?usp=drive_link
10	PCAcálculo	UNIVALLE	Colombia	Gratis, descargable	PCA	2015	Interfaz amigable y fácil de usar	No ofrece ayudas de cálculo, no tiene bases de datos.	Sin bases de datos, solo permite dos tipos de materiales como apoyo de la losa, subbase granular y subbase estabilizada con cemnto.	https://sites.google.com/a/correounivalle.edu.co/pcaindustrial/pcac%C3%A1lculo
11	BS-PCAA	Topo3 y UNICAUCA	Colombia	Gratis, descargable	PCA y AASHTO	2022	Ofrece ayuda para el cálculo de Kapoyo para base granular y base estabilizada con cemento.	Solo maneja base granular y base estabilizada con cemento, no tiene bases de datos de tránsito. En la metodología AASHTO no maneja el módulo resiliente afectado por el factos ambiental.	Sin bases de datos, Se apoya en el software Tránsito, pero no es explícto como obtener el espectro de carga para alimentar el software.	https://www.topo3.com/descargas/
11.1	Tránsito	Topo3 y UNICAUCA	Colombia	Gratis, descargable	Cuarta Potencia	2022	Base de datos de tránsito de Colombia	Requiere ser procesada para ser utilizada en el software BS-PCA, PCA, IMT-PAVE, método AASHTO o cualquier software norteamericano.	No aplica para las metodologías racionales.	https://www.topo3.com/descargas/
11.2	Clima	Topo3 y UNICAUCA	Colombia	Gratis, descargable	MTI	Sep 28 de 2022	Base de datos climatológica de Colombia.	Requiere ser procesada para ser utilizada en el software BS-PCA, PCA, IMT-PAVE, método AASHTO o cualquier software norteamericano.	Bases de datos de Colombia, Excelente, tiene conexión a las bases de datos del IDEAM.	https://www.topo3.com/descargas/

2.2 Formulación del problema

¿Se puede mejorar el proceso de diseño de pavimentos rígidos para Colombia desarrollando un prototipo de software, alimentada con bases de datos de tránsito colombianas?

2.3 Sistematización del problema

- ¿Cuál es la información relevante sobre los desafíos que enfrenta alguien al diseñar un pavimento en Colombia actualmente?
- ¿Cuál es el diseño de software adecuado para luego implementar un software que soluciones el problema?
- ¿Cómo crear una implementación del diseño de software que dé como resultado un producto mínimo viable?
- ¿Cómo realizar pruebas con usuarios y observar que tan útil es el software?

3. OBJETIVOS

3.1 Objetivo general

Desarrollar un prototipo de software para diseños de pavimentos rígidos, alimentado con bases de datos de tránsito colombianas, que facilite el uso en un contexto local y regional.

3.2 Objetivos específicos

- Reconocer la información relevante sobre los desafíos que enfrenta alguien al diseñar un pavimento en Colombia actualmente.
- Diseñar un software para esta problemática específica junto con una interfaz sencilla.
- Implementar el diseño de software que dé como resultado un producto mínimo viable.
- Realizar pruebas de usabilidad y de aceptación.

4. JUSTIFICACIÓN

Este proyecto de investigación gira sobre la necesidad de lograr que personas interesadas en el diseño de pavimentos rígidos en Colombia tengan una herramienta para optimizar el flujo de trabajo de como actualmente se realizan los diseños locales.

Este Trabajo de Grado pretende aportar con el diseño y desarrollo de un prototipo de software que ayude con los problemas identificados en la etapa de toma de requerimiento tales como el tránsito de diseño de pavimentos que incluye el cálculo de volumen de tránsito, la composición vehicular y los espectros de carga regionales y locales. El software dará la posibilidad de pasar datos de pesajes de camiones registrados en un archivo de Excel y procesarlos para obtener el espectro de carga y también presentara una base de datos de espectros de carga regionales los cuales le sirven al usuario para agilizar el flujo de trabajo y entregar diseños contextualizados.

A su vez, este proyecto podrá servir como base para un desarrollo futuro, creando otros módulos útiles.

Este proyecto se justifica porque se identifica un problema actual en la eficiencia del flujo de la adquisición de datos de tránsito para el diseño de pavimentos rígidos y de cómo esto puede llevar a problemas en los resultados de las construcciones de vías. Con esto se aporta en el proceso de diseño, volviéndolo más eficiente y confiable con el fin de alcanzar de manera indirecta el mejoramiento de la vida útil real de las vías, pues se diseñará con datos de tránsito regionales en vez de datos extranjeros.

4.1 Delimitación y alcance del proyecto

Los resultados y conclusiones del presente trabajo buscan el mejoramiento del acceso al conocimiento para diseñadores de pavimentos rígidos con caso de estudio Colombia.

Se identifico el tipo de tránsito en Colombia incluyendo los registros de pesajes realizados desde 1996 hasta la fecha en la red nacional de carreteras. Se favorecerá con esto el diseño de pavimentos en la nación.

Se trabajó con espectros de cargas colombianos, pero se brinda también la posibilidad de utilizar espectros de carga personalizados.

Respecto al diseño de pavimentos rígidos, únicamente se implementó la metodología de la PCA.

4.2 Entregables

A partir de este Trabajo de Grado se desarrolló un prototipo de software para el diseño de pavimentos rígidos con la metodología de la PCA.

El software es una aplicación web donde los usuarios pueden crear, almacenar y consultar sus diseños de pavimentos.

4.3 Área de aplicación del proyecto

De acuerdo con los SIG Listing (ACM, n.d.) de ACM este proyecto se enmarca en:

- SIGMIS - Special Interest Group on Management Information Systems.
- SIGMOD - Special Interest Group on Management of Data.
- SIGPLAN - Special Interest Group on Programming Languages.
- SIGSOFT - Special Interest Group on Software Engineering.

5. MARCO DE REFERENCIA

5.1 Principios del análisis estructural de pavimentos rígidos

Las vías carreteras conectan lugares de interés para los pobladores y su función es entregar confort y seguridad a largo plazo para los vehículos, de tal forma que se debe controlar el detrimento del pavimento garantizando que el suelo y los materiales no se deterioren por efectos del tránsito y del clima. La modelación de pavimentos rígidos sigue la ley de la elasticidad propuesta por Hooke, si ocurre que los esfuerzos o las deformaciones inducidos superan la capacidad de los materiales se presenta plastificación, fluencia y transmisión de esfuerzos al material vecino, las consideraciones iniciales de elasticidad dejan de ser correctas y si las cargas de tránsito aumentan la zona plástica y el modelo de diseño pierde validez (Juárez Badillo & Rico Rodríguez, 1991).

El pavimento rígido se compone de losas de concreto que típicamente se apoyan sobre el conjunto suelo - subbase. De acuerdo con la teoría de Westergaard los esfuerzos principales ocurren en el borde de las losas de concreto mientras que las deformaciones principales ocurren en las esquinas (Griffiths & Thom, 2007) por lo cual las losas de pavimento se diseñan bajo dos criterios, análisis de fatiga: controla los esfuerzos máximos a través del cálculo de deflexiones a causa de las repeticiones de las cargas de tránsito y análisis de erosión: controla las deformaciones en la esquina de la loza utilizando el *trabajo* producido por la acción de las cargas.

La metodología de la PCA (Portland Cement Association) se basa en la mecánica de materiales, casi en su totalidad, pero contiene algunos modelos empíricos que permiten ajustar estas teorías al verdadero comportamiento de los pavimentos ya en operación (CivilWeb Consulting Engineers, 2024). El criterio para diseñar el espesor del pavimento es limitar el número de repeticiones de carga del tránsito controlando la fatiga en el borde de las losas y la erosión en las esquinas, ocasionadas ambas por las repeticiones de las cargas del tránsito y los esfuerzos térmicos (Lee & Carpenter, 2001). El fin del diseño es evitar la iniciación de la primera grieta debido a esfuerzos críticos en los bordes y prevenir fallas del pavimento en las juntas debido a la erosión del material bajo losa de concreto (Lee & Carpenter, 2001).

La PCA utilizó el modelo de elementos finitos para calcular los esfuerzos críticos y las deflexiones en las diferentes capas del pavimento. Dichas soluciones se contrastaron con las ecuaciones desarrolladas en periodos anteriores por Westergaard (carga en el borde de la losa, alabeo por cambio de temperatura) y Bradbury (alabeo por cambio de temperatura) y con datos de campo, lo cual dio origen a las tablas y gráficas que actualmente se usan en la metodología de la PCA para realizar los diseños (Meier, 2024).

5.2 Conversión de unidades

Todas las ecuaciones del método PCA están en sistema inglés, por lo tanto, los cálculos internos del software debes hacerse en ese sistema, sin embargo, es muy posible que la entrada y la salida de datos se realice en otros sistemas para lo cual se utilizan las conversiones registradas en la Tabla 2.

Tabla 2. Conversión de unidades (Portland Cement Association (PCA), 1995)

Sistema Inglés	Otros sistemas	Se utiliza en
1.0 Pulg	25.4 mm	Espesor de la losa de concreto (h)
1.0 pies	0.305 m	Espesor de la subbase (hsb)
1.0 psi	0.00689 MPa	Resistencia a compresión (f'c) Resistencia a flexión (S'c) Módulo de elasticidad (Ec)
1.0 pci	0.271 MPa/m	Módulo de reacción de la subrasante (K) Módulo de reacción del apoyo (Kapoyo)
1.0 lb	0.454 kg	Espectro de carga
1.0 lb	4.45 N	
1.0 kip	0.454 t	
1.0 kip	4.45 kN	

5.3 Generalidades

Para el diseño de pavimentos rígidos utilizando la metodología de la PCA se requiere conocer las siguientes variables:

5.3.1 Módulo de reacción de la subrasante o suelo, K.

La subrasante es la capa donde se apoya la estructura del pavimento. Puede obtenerse a partir de diferentes estudios de laboratorio como el California Bearing Ratio (CBR), el módulo resiliente (Mr) o el módulo de reacción (K), o correlaciones

entre ellas. La Ecuación 1 (FHWA, 2006) muestra la correlación entre Mr (psi) y K (pci) y las Ecuaciones 2 y 3 muestran la correlación entre CBR (%) y K (pci). Para el correcto diseño debe utilizarse el valor de Mr afectado por el factor ambiental (AASHTO, 1993) o bien utilizar CBR sumergido (Portland Cement Association (PCA), 1995).

$$K = Mr/19.4 \qquad \text{Ecuación 1.}$$

$$K = (2.55 + 52.5 * \log(CBR))/0.271 \;\; para\; CBR < 10\% \qquad \text{Ecuación 2.}$$

$$K = (46 + 9.08 * (\log\; CBR)^{4.34})/0.271 \;\; para\; CBR \geq 10\% \qquad \text{Ecuación 3}$$

5.3.2 Módulo de reacción del apoyo, Kapoyo.

Se debe escoger el tipo de subbases (granular o estabilizada con cemento o asfalto) y el espesor de la subbase en pulg (hsb). El conjunto subrasante-subbase permite calcular el Kapoyo (Tablas 3 y 4) (CivilWeb Consulting Engineers, 2024).

Tabla 3. Efecto de la subbase granular en los valores de K

Valor de K de la subrasante [pci]	**Subbase Granular - Kapoyo [pci]**			
	Espesor [pulg]			
	4	**6**	**9**	**12**
50	65	75	85	110
100	130	140	160	190
200	220	230	270	320
300	320	330	370	430

Tabla 4. Efecto de la subbase estabilizada en los valores de K

Valor de K de la subrasante [pci]	**Subbase tratada con cemento - Kapoyo (pci)**			
	Espesor [pulg]			
	4	**6**	**8**	**10**
20	170	230	310	390
40	280	400	520	640
60	470	640	830	

5.3.3 Espesor de la losa de concreto, h (pulg).

Esta es una de las incógnitas a resolver a través del diseño y típicamente está entre 150 mm y 300 mm (ICPC, Ministerio de Transporte, INVIAS, 2008).

5.3.4 Resistencia a la flexión del concreto, S'c (psi).

Esta es otra de las incógnitas a resolver a través del diseño. Las losas de concreto de los pavimentos funcional a flexión y típicamente esta resistencia está entre 3.2 MPa y 6.0 Mpa (ICPC, Ministerio de Transporte, INVIAS, 2008).

5.3.5 Resistencia a la compresión del concreto, f'c (psi).

Los concretos de los pavimentos funcionan a flexión, pero los manuales y especificaciones para control de calidad de obra (ICPC, Ministerio de Transporte, INVIAS, 2008) sugieren realizar el ensayo de resistencia a compresión para tener correlaciones entre f'c y S'c que presentan la forma de la Ecuación 4, donde A es la constante de correlación. En caso de no tener correlación puede usarse la sugerida por la (Asociación Colombiana de Ingeniería Sísmica, 2010) (Ecuación 5).

$$S'c = A * \sqrt{f'c}$$ Ecuación 4.

$$S'c = 0.62 * \sqrt{f'c}$$ Ecuación 5.

5.3.6 Módulo de elasticidad, Ec (psi).

Se define como la relación entre el esfuerzo axial aplicado a un cuerpo y la respectiva deformación unitaria. En la Tabla 5 se encuentran algunas correlaciones para su cálculo, sin embargo, de acuerdo con la Norma Sismo Resistente de Colombia en el numeral C.8.5.1 el módulo de elasticidad puede calcularse de acuerdo con la Ecuación 6, donde Wc = Densidad, para valores comprendidos entre 1400 y 2560 Kg/m3 y f'c está en MPa. Para concreto de densidad normal (Wc ≈ 2300 Kg/cm3) con la Ecuación 7 (Asociación Colombiana de Ingeniería Sísmica, 2010).

Tabla 5. Correlación entre la resistencia a la compresión y el módulo de elasticidad. (ICPC, Ministerio de Transporte, INVIAS, 2008)

Tipo de agregado de origen	Módulo de elasticidad (MPa)
Grueso - ígneo	$Ec = 5500\sqrt{f'c}$
Grueso - Metamórfico	$Ec = 4700\sqrt{f'c}$
Grueso - Sedimentario	$Ec = 3600\sqrt{f'c}$
Sin información	$Ec = 2900\sqrt{f'c}$

$$Ec = Wc^{1.5} * 0.043\sqrt{f'c}$$ Ecuación 6.

$$Ec = 4700\sqrt{f'c}$$ Ecuación 7.

5.3.7 Relación de Poisson, ϑ.

Es una característica del concreto que se obtiene generalmente a partir del ensayo de compresión de cilindros en el laboratorio. Es la relación entre el ensanchamiento y el acortamiento del cilindro bajo la acción de una carga. Por defecto se toma $\vartheta = 0.15$ pero puede ser otro valor reportado por el laboratorio (Portland Cement Association (PCA), 1995).

5.3.8 Radio de rigidez relativa l.

Al conjunto subrasante-subbase se le denomina apoyo, porque sirven de apoyo a la losa de concreto. La relación existente entre las características de la losa de concreto y la superficie de apoyo está dada por el radio de rigidez relativo l (Ecuación 8) (Griffiths & Thom, 2007).

$$l = \sqrt[4]{\frac{Ec*h^3}{12(1-\vartheta^2)*Kapoyo}}$$ Ecuación 8.

Donde: *Ec* = Módulo de elasticidad en psi (4’000.000 psi por defecto)

ϑ = Relación de Poisson (0.15 por defecto)

Kapoyo = Módulo de reacción de la superficie de apoyo en pci.

h = Espesor de la losa de concreto en pulg.

5.3.9 Factor de seguridad de carga, FSC

Este factor se encarga de mayorar las cargas del tránsito y depende de la importancia de la via. Se recomienda los siguientes (Portland Cement Association (PCA), 1995):

- FSC = 1.3, este factor se justifica cuando se requiere mantener una serviciabilidad mayor de la normal a lo largo del periodo de desempeño de la

estructura. Se recomienda para vías de alto volumen de tránsito que no cuentan con vías alternas.

- FSC = 1.2, para vías con alto volumen de tránsito de camiones, vías interestatales y vías multicarril.
- FSC = 1.1, para vías arterias con moderado tránsito de camiones.
- FSC = 1.0, para carreteras, vías residenciales y otras calles con bajo tránsito de camiones.

5.3.10 Espectro de carga

Se refiere a la distribución de carga que transportan los camiones diferenciada por tipo de eje: simple-simple (sencillo), simple-doble (dual), tándem y trídem y tienen la forma presentada en las Figuras 1 a 4. El espectro de carga consiste en el análisis estadístico (histograma de frecuencias) de los pesos de los ejes que rodarán sobre el pavimento (Instituto Mexicano de Transporte (IMT), 2022) y puede obtenerse de varias formas:

- Registro de cargas directamente desde el proyecto vial.
- Registro de cargas regionales. Espectros derivados de operativos de pesajes de vías aledañas al proyecto vial.
- Registro de carga tipo. Cuando no existe registro de carga se usan espectros de carga disponibles en bases de datos.

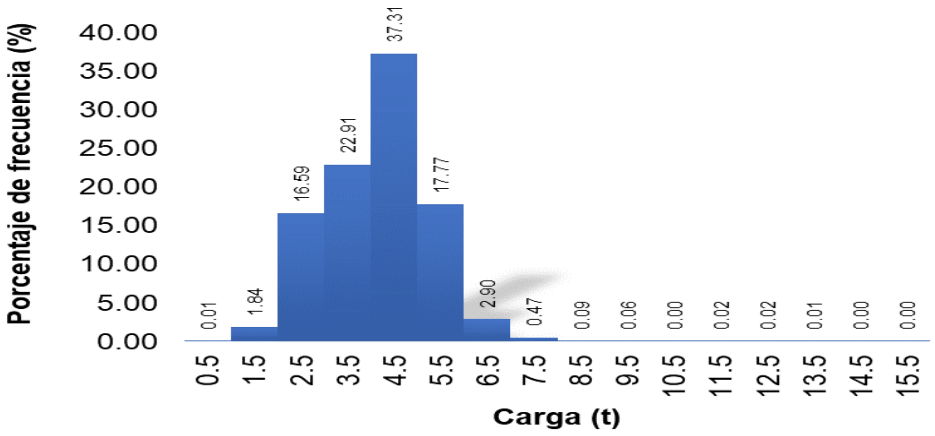


Figura 1. Histograma de frecuencia típico del eje sencillo (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

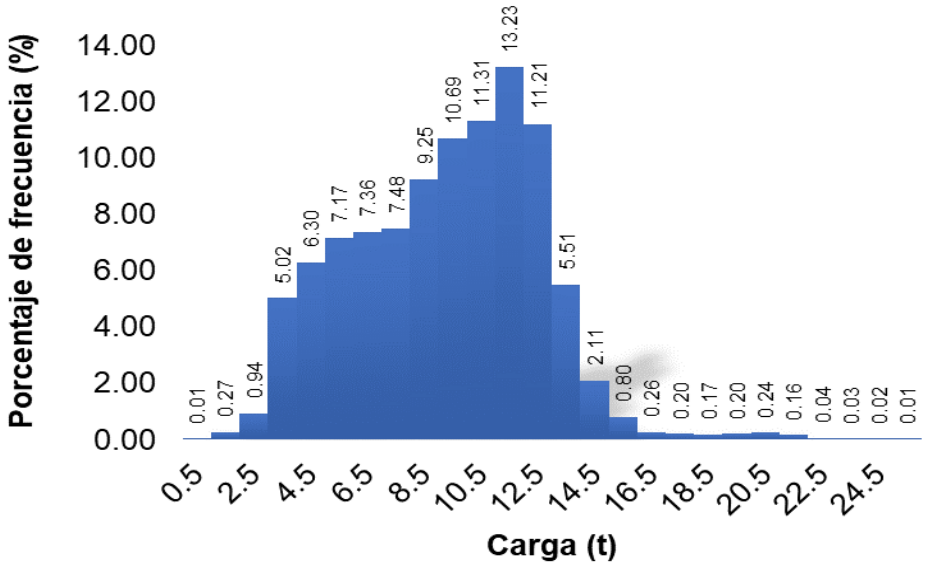


Figura 2. Histograma de frecuencia típico del eje dual (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

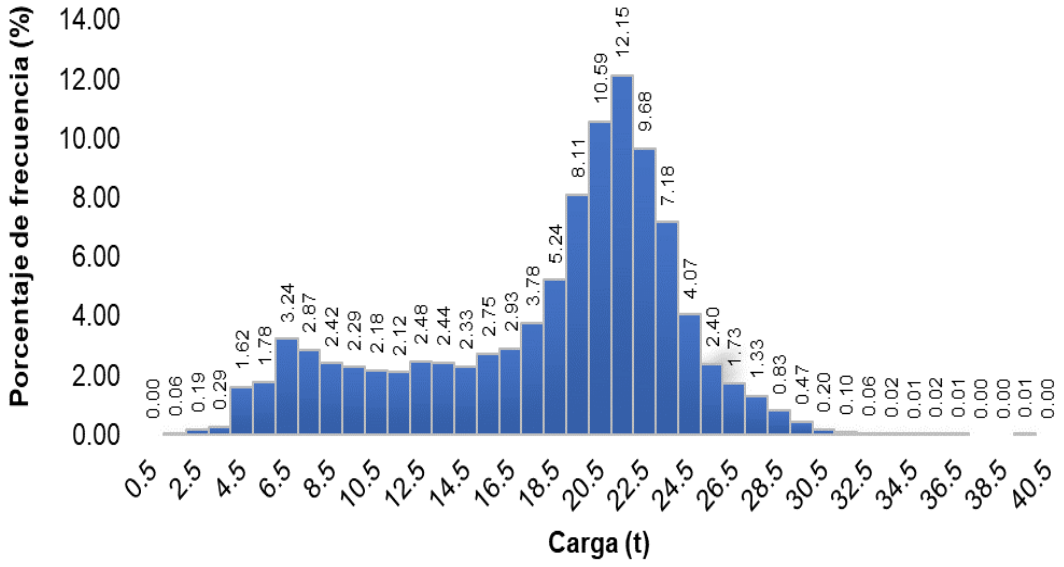


Figura 3. Histograma de frecuencia típico del eje tándem (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

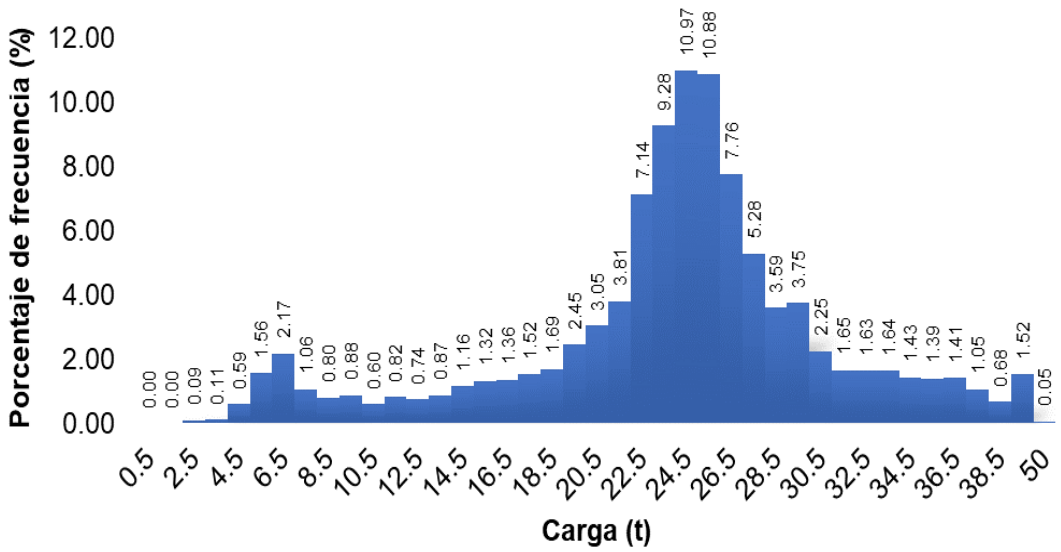


Figura 4. Histograma de frecuencia típico del eje trídem (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

5.3.11 Juntas y apoyo lateral

El pavimento de concreto no es una banda continua, sino que está conformado por losas sucesivas. Al contacto entre una losa y otra se le denomina junta y puede o no tener barras de transferencia de carga (pasadores) entre las losas contiguas. Otro aspecto importante en el diseño de las losas es la presencia de bermas o apoyo lateral (Huang, 2003).

5.3.12 Análisis de fatiga

De acuerdo con Huang, 2003 la Portland Cement Association calculó los esfuerzos sobre las losas (σ_x) con el uso del programa de elementos finitos JSLAB considerando todas las variables mencionadas en el numeral anterior. Para ello se consideró el espectro de carga y la existencia de bermas o apoyos laterales y las características de los materiales Estos resultados fueron corroborados en campo con ensayos a escala real que incluyeron los datos de comportamiento y estudios de falla de pavimentos de la prueba vial AASHTO (Huang, 2003). Los modelos de fatiga se presentan en la Ilustración 5 y los factores f1, f2, y f4 se presentan en las Figuras 5 a 7 y los Me se presentan en la Figura 8.

$$\sigma_x = \frac{6Me}{h^2} f1 * f2 * f3 * f4$$

$$\begin{cases} Si \ \left(\frac{\sigma x}{S'c}\right) \geq 0.55, & Log\ N = 11.737 - 12.077 * \left(\frac{\sigma x}{S'c}\right) \\ Si \ \ 0.45 < \left(\frac{\sigma x}{S'c}\right) < 0.55, & N = \left(\frac{4.2577}{\frac{\sigma x}{S'c} - 0.4325}\right)^{3.268} \\ Si \ \ \left(\frac{\sigma x}{S'c}\right) < 0.45, & N = Ilimitado \end{cases}$$

Figura 5. Cantidad de ejes admisibles N con relación a los esfuerzos actuantes en el borde de la losa (σx) y la resistencia a flexión del concreto (S'c) (Huang, 2003)

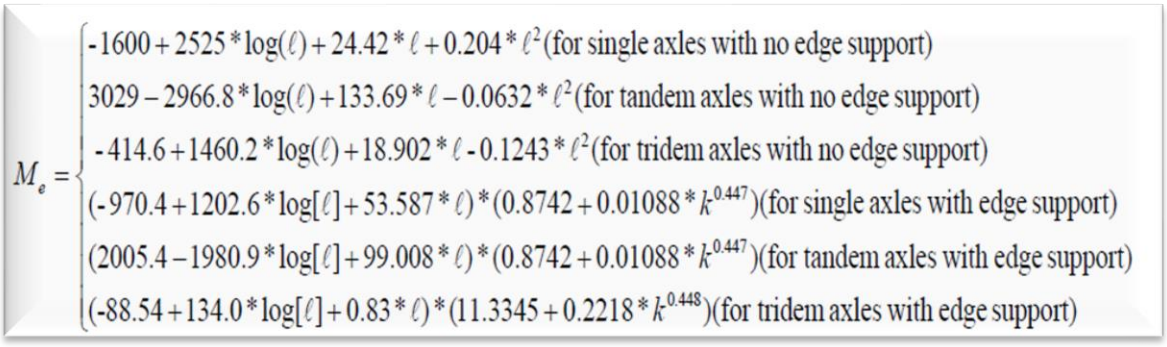
$$f_1 = \begin{cases} \left(\frac{\text{SAL}}{24}\right)^{0.94} * \frac{24}{18} \text{ for single axles} \\ \left(\frac{\text{TAL}}{48}\right)^{0.94} * \frac{48}{36} \text{ for tandem axles} \\ \left(\frac{\text{TRIAL}}{72}\right)^{0.94} * \frac{72}{54} \text{ for tridem axles} \end{cases}$$

Figura 6. Factor f1, debido al tipo de eje y presión de contacto, SAL, TAL, TRIAL son el espectro de carga por tipo de eje (Huang, 2003)

$$f_2 = \begin{cases} 0.892 + \left(\frac{h_c}{85.71}\right) - \frac{h_c^{\ 2}}{3000} \text{ for no shoulders} \\ 1 \text{ for with shoulder} \end{cases}$$

Figura 7. Factor f2, para pavimentos sin bermas en concreto (Huang, 2003)

El factor f3 = 0.894 asumen que el 6% de los camiones ruedan al borde de la losa y el factor f4 es $1/[1.235 * (1 - CV)]$, siendo CV el coeficiente de variación de f'c (Huang, 2003).

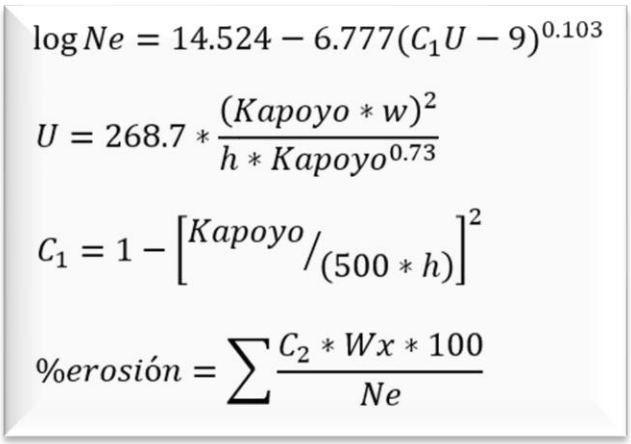


$$M_e = \begin{cases} -1600 + 2525 * \log(\ell) + 24.42 * \ell + 0.204 * \ell^2 \text{ (for single axles with no edge support)} \\ 3029 - 2966.8 * \log(\ell) + 133.69 * \ell - 0.0632 * \ell^2 \text{ (for tandem axles with no edge support)} \\ -414.6 + 1460.2 * \log(\ell) + 18.902 * \ell - 0.1243 * \ell^2 \text{ (for tridem axles with no edge support)} \\ (-970.4 + 1202.6 * \log[\ell] + 53.587 * \ell) * (0.8742 + 0.01088 * k^{0.447}) \text{ (for single axles with edge support)} \\ (2005.4 - 1980.9 * \log[\ell] + 99.008 * \ell) * (0.8742 + 0.01088 * k^{0.447}) \text{ (for tandem axles with edge support)} \\ (-88.54 + 134.0 * \log[\ell] + 0.83 * \ell) * (11.3345 + 0.2218 * k^{0.448}) \text{ (for tridem axles with edge support)} \end{cases}$$

Figura 8. Momento equivalente, Me, para eje simple (single), tándem y trídem con y sin apoyo lateral (Edge support) (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

5.3.13 Análisis de erosión

La erosión en los pavimentos rígidos se da en las juntas, que es el sitio por donde puede acceder agua hasta la superficie de apoyo y generar erosión (Huang, 2003). Las ecuaciones para calcular el número de repeticiones admisibles en la esquina debido al trabajo U que realiza la losa al sufrir una deflexión w (Figura 9). El valor de w se calcula con las ecuaciones que aparecen en la Figura 10.



$$\log Ne = 14.524 - 6.777(C_1 U - 9)^{0.103}$$

$$U = 268.7 * \frac{(Kapoyo * w)^2}{h * Kapoyo^{0.73}}$$

$$C_1 = 1 - \left[{}^{Kapoyo}/_{(500 * h)} \right]^2$$

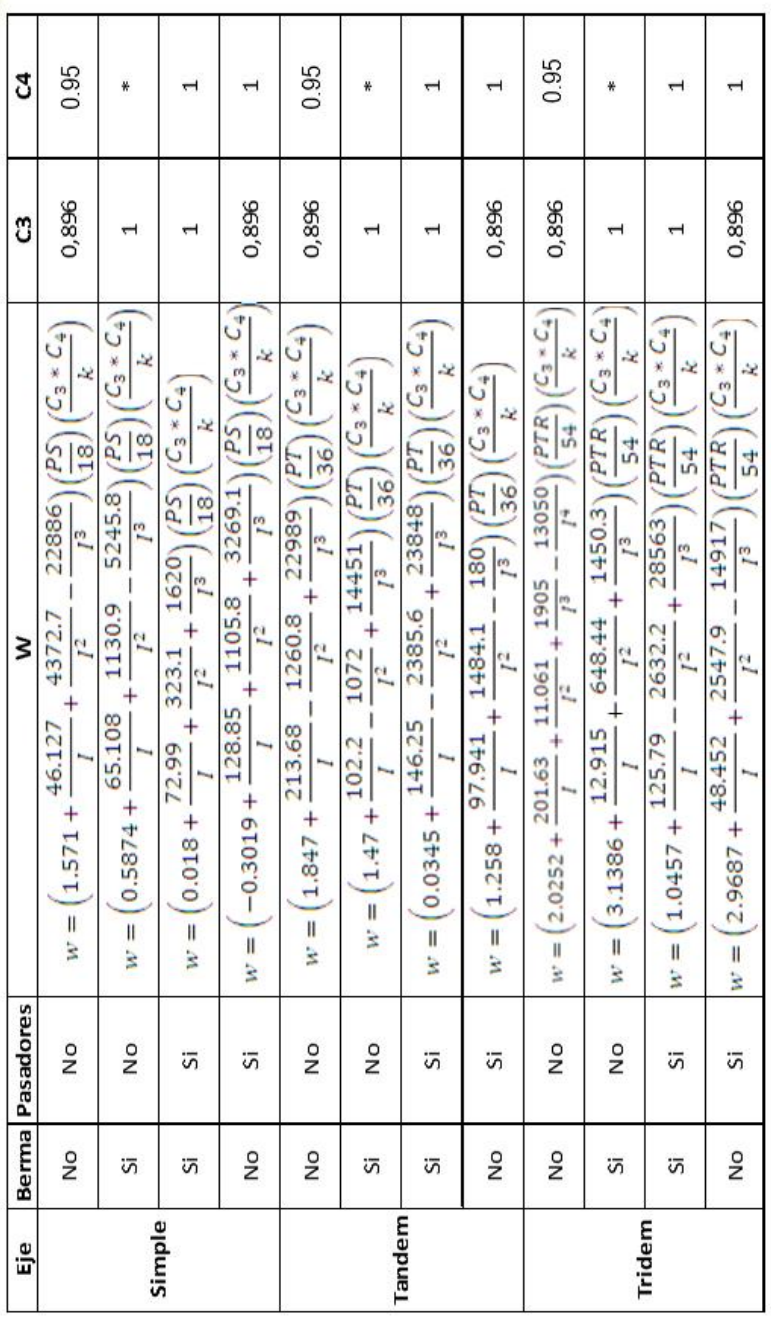
$$\%erosión = \sum \frac{C_2 * Wx * 100}{Ne}$$

Figura 9. Cantidad de ejes admisibles Ne con relación al trabajo U que realiza la losa en la esquina (Huang, 2003)

El porcentaje de daño debido a erosión, al igual que en el análisis de fatiga depende de la cantidad de repeticiones admisibles por erosión (Ne/C2) en comparación con las esperadas (Wx) (Huang, 2003)

C2 = 0.06 para pavimento sin apoyo lateral

C2 = 0.94 para pavimento con apoyo lateral



Eje	Berma	Pasadores	W	C3	C4
Simple	No	No	$w = \left(1.571 + \frac{46.127}{l} + \frac{4372.7}{l^2} - \frac{22886}{l^3}\right)\left(\frac{PS}{18}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	0.95
	Si	No	$w = \left(0.5874 + \frac{65.108}{l} + \frac{1130.9}{l^2} - \frac{5245.8}{l^3}\right)\left(\frac{PS}{18}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	*
	Si	Si	$w = \left(0.018 + \frac{72.99}{l} + \frac{323.1}{l^2} + \frac{1620}{l^3}\right)\left(\frac{PS}{18}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	1
	No	Si	$w = \left(-0.3019 + \frac{128.85}{l} + \frac{1105.8}{l^2} + \frac{3269.1}{l^3}\right)\left(\frac{PS}{18}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	1
Tandem	No	No	$w = \left(1.847 + \frac{213.68}{l} - \frac{1260.8}{l^2} + \frac{22989}{l^3}\right)\left(\frac{PT}{36}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	0.95
	Si	No	$w = \left(1.47 + \frac{102.2}{l} - \frac{1072}{l^2} + \frac{14451}{l^3}\right)\left(\frac{PT}{36}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	*
	Si	Si	$w = \left(0.0345 + \frac{146.25}{l} - \frac{2385.6}{l^2} + \frac{23848}{l^3}\right)\left(\frac{PT}{36}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	1
Tridem	No	Si	$w = \left(1.258 + \frac{97.941}{l} + \frac{1484.1}{l^2} - \frac{180}{l^3}\right)\left(\frac{PT}{36}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	1
	No	No	$w = \left(2.0252 + \frac{201.63}{l} + \frac{11.061}{l^2} + \frac{1905}{l^3} - \frac{13050}{l^4}\right)\left(\frac{PTR}{54}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	0.95
	Si	No	$w = \left(3.1386 + \frac{12.915}{l} + \frac{648.44}{l^2} + \frac{1450.3}{l^3}\right)\left(\frac{PTR}{54}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	*
	Si	Si	$w = \left(1.0457 + \frac{125.79}{l} - \frac{2632.2}{l^2} + \frac{28563}{l^3}\right)\left(\frac{PTR}{54}\right)\left(\frac{C_3 * C_4}{k}\right)$	1	1
	No	Si	$w = \left(2.9687 + \frac{48.452}{l} + \frac{2547.9}{l^2} - \frac{14917}{l^3}\right)\left(\frac{PTR}{54}\right)\left(\frac{C_3 * C_4}{k}\right)$	0,896	1

$C_3 = Factores\ de\ ajuste\ por\ camión\ en\ el\ borde\ del\ pavimento$

$C_4 = Factores\ de\ ajuste\ de\ forma\ de\ transmisión\ de\ carga\ en\ transversales$

$* C_4 = F = 1.001 - \left(0.26363 - \frac{k}{3034.5}\right)^2$

Figura 10. Deflexiones w para el espectro de carga de cada tipo de eje y en diversas condiciones de pasadores y apoyo lateral. Léase k como Kapoyo (García Aladín, Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas, 2002)

6. MARCO TEÓRICO

6.1 Ingeniería de Software

La ingeniería de software abarca todo el ciclo de vida del desarrollo de software, desde la concepción de la idea hasta el mantenimiento y la evolución del producto. Esto incluye la aplicación de principios de ingeniería para gestionar de manera efectiva los recursos, el tiempo y el personal involucrado en el proyecto, así como para garantizar la calidad del software resultante. (Pressman, 2015) (Sommerville, 2016).

El desarrollo del software se lleva a cabo aplicando los principios fundamentales de la ingeniería de software, que abarcan todo el ciclo de vida del desarrollo. Desde la concepción de la idea, se utiliza una metodología que permita gestionar eficazmente los recursos, el tiempo y el equipo involucrado, lo que asegura que el producto final cumpla con los estándares de calidad y funcionalidad esperados.

El prototipo del software se diseña considerando no solo las necesidades inmediatas del proyecto, sino también la escalabilidad y el mantenimiento a largo plazo. Se establecen procesos para asegurar la correcta documentación, pruebas y validación del software, lo que permite su evolución conforme a las necesidades del sector. Además, la planificación y el control del desarrollo están guiados por técnicas que aseguran la optimización de los recursos disponibles, garantizando que el software se entregue a tiempo y dentro del presupuesto.

El uso de prácticas de ingeniería de software ayuda a gestionar de manera eficiente los riesgos y a implementar mecanismos de control de calidad, como la revisión de código, pruebas unitarias y de integración. Esto garantiza que el prototipo sea técnicamente sólido, fácil de usar y adaptado a las condiciones locales.

6.2 Metodología Ágil

En este numeral se hace una breve descripción general de la metodología Agile, explicando sus principios, ventajas y relevancia para el desarrollo del sistema de gestión de diseño de usuarios. Se eligió la metodología Agile para este proyecto con el fin de facilitar el desarrollo iterativo, mejorar la flexibilidad y garantizar la entrega continua de software funcional.

Las metodologías ágiles son enfoques iterativos e incrementales para el desarrollo de software que priorizan la adaptabilidad, la colaboración y la entrega

temprana de software funcional. Los equipos ágiles trabajan en ciclos cortos llamados iteraciones o sprints, donde se desarrollan, prueban y entregan pequeñas partes del producto en intervalos regulares. (Manifesto, 2001) (Manifesto for Agile Software Development, 2001).

El desarrollo del software de diseño se lleva a cabo utilizando metodologías ágiles, que permitan una evolución iterativa e incremental del prototipo. Esto garantiza la entrega de partes funcionales del software en intervalos regulares, promoviendo la adaptabilidad y facilitando la incorporación de mejoras continuas en respuesta a los resultados obtenidos y la retroalimentación recibida.

El equipo de trabajo, en este caso conformado por un estudiante de ingeniería de sistemas y computación, el docente director del proyecto (ingeniero de sistemas y computación), y una ingeniera civil experta en pavimentos, colaboran estrechamente durante cada ciclo de desarrollo. Cada miembro del equipo aporta su experiencia específica, asegurando que tanto los aspectos técnicos del software como los requerimientos del diseño de pavimentos sean considerados de manera integral.

Durante cada sprint, se desarrollan, prueban y validan funcionalidades del software, enfocándose en la entrega temprana de componentes esenciales. Este enfoque no solo permite ajustes rápidos y oportunos, sino que también facilita la validación continua del prototipo por parte de los expertos, alineando el software con los objetivos del proyecto y las expectativas de los usuarios finales.

6.2.1 Principios de la metodología Agile

La metodología Agile se basa en los principios delineados en el Manifiesto Ágil, que enfatiza a los individuos y las interacciones, el software funcional, la colaboración con el cliente y la respuesta al cambio (Beck, et al., 2001).

- **Individuos e interacciones:** Priorizar la comunicación y la colaboración entre los miembros del equipo por encima de procesos y herramientas rígidos.
- **Software:** Centrarse en entregar software funcional con frecuencia, con énfasis en lanzamientos pequeños e incrementales.
- **Colaboración con el cliente:** Involucrar a las partes interesadas y a los clientes en el proceso de desarrollo para garantizar que el producto final satisfaga sus necesidades.
- **Respuesta al cambio:** Aceptar cambios en los requisitos, incluso en etapas avanzadas del proceso de desarrollo, para mejorar el valor y la relevancia del producto.

6.2.2 Beneficios de la metodología Agile

- **Flexibilidad y adaptabilidad**: La metodología ágil permite la reevaluación y adaptación frecuente de los planes, lo que resulta crucial para proyectos con requisitos cambiantes. Esta flexibilidad garantiza que el proyecto pueda adaptarse a nuevos conocimientos o cambios en las necesidades de los usuarios (Schwaber & Sutherland, 2017).
- **Entrega continua y feedback (retroalimentación)**: Las prácticas ágiles promueven la entrega continua de pequeños incrementos funcionales del software. Este enfoque permite recibir comentarios periódicos de las partes interesadas y los usuarios, lo que permite realizar ajustes y mejoras oportunos (Ries, 2011).
- **Colaboración mejorada**: Al fomentar una colaboración estrecha entre desarrolladores, partes interesadas y usuarios, Agile garantiza que todas las partes estén alineadas y contribuyan activamente al éxito del proyecto. Este entorno colaborativo mejora la comunicación y reduce los malentendidos (Highsmith, 2009).
- **Calidad mejorada**: Las pruebas frecuentes y la integración continua son parte integral de las prácticas ágiles, lo que conduce a un software de mayor calidad. Estas prácticas ayudan a identificar y abordar los defectos en las primeras etapas del proceso de desarrollo, lo que reduce el riesgo de problemas importantes más adelante (Rubin, 2012).
- **Relevancia para el sistema**: La adopción de la metodología Agile para el desarrollo del sistema de gestión de diseño de usuarios se alinea con la necesidad del proyecto de flexibilidad y progreso iterativo. Dado que el proyecto involucra a múltiples usuarios y características que pueden evolucionar, el enfoque iterativo de Agile garantiza que el sistema pueda adaptarse a los requisitos cambiantes y a los comentarios de los usuarios de manera eficaz.
- **Desarrollo iterativo:** La naturaleza iterativa de Agile permite el desarrollo incremental de funciones, lo que permite que el sistema crezca y evolucione en respuesta a las necesidades y los comentarios de los usuarios.
- **Participación de las partes interesadas:** La colaboración continua con las partes interesadas garantiza que el producto final se alinee con sus expectativas y requisitos, mejorando la satisfacción del usuario.

6.3 Diseño de Arquitectura

El diseño de arquitectura de software se refiere a la estructuración y organización de un sistema de software en componentes y subsistemas, así como a las interacciones entre ellos. Esto implica tomar decisiones sobre la distribución de la funcionalidad, la asignación de responsabilidades y la definición de interfaces entre los componentes del sistema. (Rozanski, 2011) (Bass, 2013).

La arquitectura de software para el desarrollo del sistema de diseño de pavimentos rígidos es un aspecto fundamental en la creación de una aplicación web eficiente, escalable y mantenible. El diseño arquitectónico del sistema implica la estructuración del software en componentes claramente definidos, donde cada uno tiene responsabilidades específicas y funciones bien delimitadas.

En términos de organización, el sistema se puede dividir en tres capas principales: la capa de presentación (interfaz de usuario), la capa de lógica de negocio (donde se aplicó la metodología PCA y se procesan los datos de tránsito, suelos, materiales y cálculo de esfuerzos y deformaciones), y la capa de datos (gestión y almacenamiento de los espectros de carga y otros datos relacionados). Estas capas interactúan entre sí a través de interfaces bien definidas, lo que permite el modularidad del sistema y la posibilidad de realizar mejoras o cambios en una capa sin afectar el resto del sistema.

Al adoptar este enfoque de arquitectura de software, se toman decisiones sobre la distribución de la funcionalidad entre los componentes, asignando la responsabilidad de los cálculos complejos a la lógica de negocio y asegurando que la interfaz de usuario ofrezca una experiencia fluida y accesible para los ingenieros que utilizarán el software. Además, la definición clara de **interfaces** entre los componentes garantiza que cada subsistema pueda interactuar de manera eficiente y consistente.

6.4 Diseño de Endpoints

El diseño de endpoints se centra en la definición y diseño de los puntos de acceso o interfaces a un sistema de software, a través de los cuales se pueden realizar solicitudes y acceder a los recursos del sistema. Los endpoints pueden ser URLs en una API web, funciones en un servicio web o métodos en una interfaz de programación. (Richardson, 2018) (Pautasso).

El diseño de endpoints en la aplicación web es un componente esencial para garantizar la interacción eficiente y segura entre el usuario y el sistema. Los endpoints se definen como URLs específicas dentro de una API RESTful, a través de las cuales los usuarios pueden autenticarse, enviar datos y recibir resultados de los cálculos de diseño de pavimentos.

Cada usuario se autentica mediante un endpoint de inicio de sesión con su nombre de usuario y contraseña. Una vez autenticado, el sistema proporciona acceso a su perfil personal, donde se almacenan todos los datos de diseño relacionados con los proyectos de pavimentos rígidos que ha creado. Los endpoints gestionan las

operaciones de creación, lectura, actualización y eliminación (CRUD) de estos proyectos, permitiendo al usuario consultar y editar los datos en cualquier momento.

6.5 Diseño de Bases de Datos

El diseño de bases de datos implica la definición de la estructura y organización de una base de datos para almacenar y gestionar la información de manera eficiente y segura. Esto incluye la identificación de las entidades y relaciones del dominio, la normalización de datos, la definición de esquemas y la optimización del rendimiento. (Connolly, 2014) (Elmasri, 2015). En la aplicación web es un componente clave para almacenar y gestionar la información relacionada con los proyectos de pavimentos rígidos de manera eficiente, segura y escalable.

6.6 Diseño de Interfaz de Usuario

El diseño de interfaz de usuario (UI) se centra en la presentación visual y la interacción del usuario con un sistema de software. Esto implica diseñar la disposición de elementos en pantalla, la navegación entre pantallas, la selección de colores y tipografías, y la creación de controles y elementos de interacción intuitivos y fáciles de usar. (Shneiderman).

El diseño de la interfaz de usuario (UI) se centra en ofrecer una presentación visual sobria y profesional, alineada con la naturaleza técnica del software y las necesidades de los usuarios. El objetivo principal es facilitar la navegación y la entrada de datos sin distracciones, permitiendo que los ingenieros que utilicen el sistema puedan concentrarse en sus proyectos de diseño de pavimentos.

7. ANÁLISIS DE REQUERIMIENTOS

7.1 Perfil de la metodología PCA

La metodología requiere de estudios de suelos, transito, materiales de construcción, entre otros, que se procesan de acuerdo con los modelos matemáticos creados por la PCA para el diseño de pavimentos rígidos.

En método inicia con la petición de las variables de diseño, es decir el tránsito, el suelo, materiales de subbase, características de la losa de concreto y factor de seguridad de carga. De cada una de las variables se debe mostrar las opciones para la adquisición de datos. Por ejemplo, el suelo puede caracterizarse en el laboratorio a partir del ensayo de CBR% o del ensayo de placa (K), de donde el software debe permitir al usuario ingresar alguna de estas variables (CBR% o K).

Respecto a los materiales de subbase, el diseñador debe seleccionar entre subbase granular o algún tipo de subbase estabilizada y dar un espesor de esa capa. El software internamente debe combinar las características del suelo con las de la subbase para producir el valor característico de la superficie de apoyo de la losa, al cual se le denomina K-apoyo.

Para la losa de concreto se debe pedir la resistencia flexión (S'c) y el espesor de la losa (h).

En la variable de tránsito se debe solicitar el transito promedio diario (TPD), la composición vehicular y el histograma de frecuencia de cargas por tipo de eje (simple, dual, tándem o trídem). De acuerdo con estos datos el software debe calcular el espectro de carga. El factor de seguridad de carga depende del TPD o de la importancia de la vía, el usuario debe poder seleccionar un valor entre 1.0 y 1.3.

El diseñador debe poder seleccionar el uso de pasadore y bermas, ya que los modelos matemáticos están condicionados a estas variables.

Una vez se han ingresado todas las variables el software debe calcular los esfuerzos en el centro de la losa (fatiga) y las deformaciones en la esquina de la losa (erosión), a esto se le denomina consumo. Ambos consumos deben estar por debajo del 100%, el usuario debe poder modificar su propuesta de diseño hasta cumplir este requisito.

En el **Anexo 1** se sintetiza gráficamente todo el proceso de diseño. A continuación, se detalla la filosofía del método.

7.1.1 Estudio de suelos

El método de diseño PCA se basa en gran medida en el módulo de reacción de la subrasante (valor k) como parámetro de entrada clave. Por lo tanto, la determinación precisa del valor k mediante pruebas de campo y de laboratorio es esencial para garantizar la adecuación estructural del pavimento y minimizar el riesgo de fallas prematuras o costos excesivos de mantenimiento (Shaheen, Tayabji, & Mounanga, 2022); (Tayabji, Buch, & Wojakowski, 2020).

Al realizar un estudio integral del suelo e incorporar los datos de subrasante obtenidos en el proceso de diseño de PCA, los ingenieros pueden optimizar el diseño del pavimento para cumplir con los criterios de desempeño requeridos y garantizar la capacidad de servicio y la rentabilidad a largo plazo. Las propiedades del suelo de subrasante juegan un papel importante en la determinación de la capacidad estructural y el desempeño del pavimento. El estudio del suelo debe proporcionar información sobre las características de resistencia, compresibilidad y drenaje de la subrasante, para lo cual se realizan varios ensayos:

- Clasifican los suelos de acuerdo con el Sistema Unificado de Clasificación de Suelos (USCS) en gravas, arenas, limos y arcillas y combinaciones entre ellos o sistema de clasificación AASHTO (Tarefder & Ahsan, 2021).
- Medición de las propiedades del suelo como plasticidad y contenido de humedad (Rao, Rao, Barai, & Momin, 2022).
- Determinación de la densidad y compactación in situ para evaluar el grado de compactación y el potencial de asentamiento futuro (Huang, 2003).
- Determinación de la resistencia, capacidad portante y rigidez del suelo con el ensayo de California Bearing Ratio (CBR) o el Módulo Resiliente (Delatte, 2014).
- En diseños de rehabilitación de pavimentos es posible utilizar el módulo de reacción de la subrasante (valor k) mediante pruebas de campo, como pruebas de carga de placas o pruebas de deflectómetro de impacto (FWD) (Shaheen, Tayabji, & Mounanga, 2022).

7.1.2 Estudio de transito

Para el diseño de un pavimento rígido es esencial un estudio de tráfico integral para determinar con precisión la carga de tráfico anticipada sobre la estructura del pavimento. El estudio de tráfico proporciona información crítica que influye directamente en el espesor de diseño del pavimento y los requisitos de transferencia de carga. La literatura reciente destaca los componentes clave de un estudio de

tráfico que se debe realizar para el diseño de pavimento rígido utilizando el método PCA, tal como se describe a continuación:

- Análisis del volumen de tráfico incluye la estimación del Tráfico Promedio Diario Anual (TPDA) y la composición de vehículos (autos, buses y tipos de camiones) (Mallick & El-Korchi, 2018).
- Estimación de tasas de crecimiento del tráfico a futuro basadas en datos históricos, planes de desarrollo regional y pronósticos económicos (Sadeghi, Khazanovich, & Rodriguez, 2022).
- Determinar el factor de distribución direccional y de carriles de diseño, que tiene en cuenta la distribución del tráfico en múltiples carriles (Khazanovich & Tompkins, 2022).
- Medición de los pesos brutos de los vehículos y configuraciones de ejes mediante sistemas de pesaje en movimiento (WIM) o básculas portátiles (Khazanovich & Tompkins, 2022).
- Espectro de cargas por eje, incluida la distribución de cargas por eje simple, tándem y trídem (Sadeghi, Khazanovich, & Rodriguez, 2022).
- Consideración de posibles cambios en las configuraciones de los vehículos y las distribuciones de carga por eje durante la vida de diseño del pavimento (Khazanovich & Tompkins, 2022).

7.1.3 Caracterización de materiales

El método de diseño PCA depende en gran medida de las propiedades de la mezcla de concreto y otros materiales utilizados en la estructura del pavimento. La caracterización precisa de estos materiales mediante pruebas de laboratorio y evaluaciones de desempeño es esencial para determinar los parámetros de diseño apropiados, como el espesor de la losa, el espaciamiento de las juntas y los requisitos de refuerzo (Goel & Das, 2021); (Nath, Maheshwari, & Kumar, 2022). Al realizar un estudio integral de materiales e incorporar los datos obtenidos en el proceso de diseño de PCA, los ingenieros pueden garantizar que el pavimento esté construido con materiales duraderos y de alta calidad, minimizando el riesgo de fallas prematuras y maximizando la vida útil y la rentabilidad del pavimento.

7.1.4 Concreto de cemento hidráulico:

Respecto a la mezcla de concreto es importante especificar la resistencia a la compresión y a la flexión requerida para satisfacer las cargas de tráfico y las condiciones ambientales (INVIAS, 2023); (INVIAS, 2008); (Goel & Das, 2021). El diseñador debe considerar los materiales cementantes, agregados y aditivos apropiados para cumplir con los requisitos de desempeño y la disponibilidad local (Nath, Maheshwari, & Kumar, 2022). También debe considerar la resistencia del concreto a factores ambientales, como ciclos de hielo-deshielo, ataque de sulfatos y reacción álcali-sílice (ASR) (Goel & Das, 2021).

7.1.5 Bases y subbases granulares y estabilizadas:

La caracterización de bases y subbases granulares y estabilizadas es un aspecto crítico del diseño de pavimentos, ya que estas capas desempeñan un papel crucial al proporcionar soporte estructural, distribución de carga y drenaje para todo el sistema de pavimento. De acuerdo con las Especificaciones Generales de Construcción de Carreteras (INVIAS, 2023) estos materiales deben ser bien gradados, resistentes, durables, con geometría adecuada y no reactivos, tal como se describe a continuación:

7.1.6 Bases y subbases granulares:

La gradación de los materiales granular afecta las características de resistencia, capacidad portante, permeabilidad y compactación del material (Haider, Harichandran, & Buch, 2021). La resistencia al corte y la capacidad portante se evalúan mediante pruebas como el California Bearing Ratio (CBR) o el Módulo Resiliente. Representan la capacidad del material para distribuir cargas a las capas subyacentes (Maina & Luo, 2022). La permeabilidad y drenaje se estudian para garantizar el drenaje adecuado y evitar la acumulación excesiva de presión de agua en los poros, lo que puede provocar una falla prematura (Haider, Harichandran, & Buch, 2021). La compactación y densidad se realizan con pruebas de compactación en laboratorio como el Proctor modificado, se utilizan para establecer la densidad seca máxima y el contenido de humedad óptimo para lograr el nivel deseado de compactación en campo (Saha & & Hanaideh, 2021).

7.1.7 Bases y subbases Estabilizadas:

La elección de agentes estabilizadores, como cemento, cal o materiales bituminosos depende de factores como las propiedades del suelo o agregado a

estabilizar, las condiciones ambientales y las características de rendimiento deseadas (Saha & & Hanaideh, 2021). Se debe considerar la realización de pruebas de laboratorio para determinar la dosis óptima del agente estabilizante y los aditivos necesarios para lograr la resistencia, durabilidad y trabajabilidad deseadas de la subbase estabilizada (Maina & Luo, 2022). La caracterización de resistencia y rigidez depende del agente estabilizante. Las resistencias a compresión, tracción y flexión, así como el módulo elástico de la subbase estabilizada se evalúan mediante pruebas de laboratorio como resistencia a la compresión no confinada (UCS) (Haider, Harichandran, & Buch, 2021) o pruebas de estabilidad Marshall para materiales estabilizados con asfalto. La durabilidad y resistencia a los ciclos de congelación y descongelación, daños por humedad y ataques químicos (por ejemplo, ataque de sulfatos) se evalúa para garantizar el rendimiento y la durabilidad a largo plazo (Saha & & Hanaideh, 2021).

7.1.8 Construcción del espectro de carga (Prozzi & & Hong, 2006)

La metodología de cálculo del espectro de carga para el diseño de pavimentos descrita en el artículo "Highway Traffic Characterization for Pavement Structure Design" de Prozzi y Hong (2006) proporciona un enfoque integral para cuantificar las cargas de tráfico previstas en una estructura de pavimento. Esta metodología es crucial para un diseño preciso de pavimentos, ya que tiene en cuenta las diversas configuraciones de ejes y magnitudes de carga que contribuyen al daño general por fatiga y al deterioro estructural del pavimento. La metodología de cálculo del espectro de carga implica realizar estudios de tráfico y recopilar datos sobre clasificaciones de vehículos, configuraciones de ejes, pesos de ejes y volúmenes de tráfico y utilizar técnicas como sistemas de pesaje en movimiento (WIM), básculas portátiles y recuentos de tráfico manuales para obtener datos representativos.

El estudio de tránsito comprende al menos dos tipos de estudio: el de volumen y clasificación vehicular y el de pesaje de vehículos de carga. El tamaño de las muestras de estos estudios sigue el rigor estadístico para garantizar que los datos obtenidos representen adecuadamente la dinámica de transporte de carga y pasajeros de la vía.

Con el primer estudio se obtiene el flujo de tráfico el cual se proyecta a futuro considerando el crecimiento económico de la región. El total obtenido se distribuye de acuerdo con la clasificación de vehículos: autos, buses y camiones, estos últimos se clasifican camiones de una sola unidad, camiones con semirremolques y combinaciones de múltiples remolques.

Con el segundo estudio se registran los pesos de los vehículos separando cada uno de sus ejes. De la información obtenida con este proceso se puede calcular la cantidad de ejes simples, tándem y trídem. Cuando la muestra se ha recolectado

en su totalidad se construye la distribución de frecuencia o las funciones de densidad de probabilidad (PDF) que representen los patrones de carga para cada tipo de eje. Cada tipo de eje se distribuye de acuerdo con la distribución de frecuencia o la función de densidades de ese tipo de eje y se construye utilizando clases de una tonelada, es decir de 0.0 t a 1.0 t, de 1.0 t a 2.0 t y así sucesivamente. El espectro de carga representa la distribución de cargas que experimentará el pavimento durante su vida de diseño.

La metodología de cálculo del espectro de carga proporciona una representación integral de la carga de tráfico anticipada sobre la estructura del pavimento, teniendo en cuenta la diversa gama de configuraciones de ejes, magnitudes de carga y volúmenes de tráfico. Esta información es crucial para las metodologías de diseño de pavimentos, como el método de la Asociación del Cemento Portland (PCA), ya que sirve como un insumo clave para determinar el espesor requerido del pavimento y la capacidad estructural.

Al caracterizar con precisión la carga de tráfico a través de la metodología de cálculo del espectro de carga, los ingenieros pueden garantizar que el pavimento esté diseñado para soportar las cargas de tráfico previstas y mantener un rendimiento estructural adecuado durante toda su vida útil prevista.

7.2 Actores involucrados en el uso de la aplicación

- Ingenieros civiles o afines que realizan diseños de pavimentos rígidos en entornos públicos y privados
- Educadores que enseñen la asignatura de pavimentos rígidos
- Alumnos que necesiten utilizar la herramienta en el diseño de pavimentos rígidos

7.3 Perfil del usuario

Los usuarios potenciales del software de diseño de pavimentos son profesionales en el campo de la ingeniería civil, principalmente aquellos involucrados en proyectos de infraestructura vial. Estos usuarios suelen poseer un conjunto diverso de habilidades y conocimientos para utilizar eficazmente el software y garantizar el diseño de estructuras de pavimento seguras, duraderas y rentables.

Este profesional generalmente comprende de forma sólida los principios, teorías y metodologías de diseño de pavimentos. Es competente en análisis estructural, del tránsito, los materiales disponibles y los conceptos de ingeniería geotécnica relacionados con el diseño de pavimentos. Conoce las especificaciones de diseño relevantes. Analiza y evalúa alternativas de diseño y optimiza estructuras de pavimento en función de las limitaciones y requisitos específicos del proyecto.

7.4 Requerimientos de la aplicación

Se obtienen los requerimientos luego de un proceso de análisis de la problemática que se presenta en las reuniones con los interesados del desarrollo de este proyecto.

7.4.1 Requerimientos No funcionales

Seguridad:

- Las claves de los usuarios se encriptan para proteger la información del proyecto.
- Los diseños de un usuario solo pueden ser accedidos por el usuario al que pertenecen
- Los proyectos de un usuario solo pueden ser vistos por el usuario al que pertenecen

Mantenibilidad:

- El Software es desarrollado siguiendo principios de diseño para asegurar que la aplicación final pueda escalar y tener desarrollos futuros.

Portabilidad:

- El software debe ser una aplicación web que sea utilizada desde navegadores.

Confiabilidad:

- El sistema debe dar diseños basados en los modelos matemáticos aceptados en la ingeniería de pavimentos.

7.4.2 Requerimientos funcionales

La aplicación deberá permitirle al usuario:

- Interactuar con el sistema por medio de una interfaz gráfica.
- crear su usuario.
- hacer login con su usuario.
- crear proyectos nuevos.
- Nombrar y modificar nombres de proyectos.
- Crear vías/diseños dentro de un proyecto.
- Modificar los diseños dentro de un proyecto.
- Diseñar los pavimentos rígidos pertenecientes a un proyecto.
- Ver un reporte resumido de los diseños dentro de un proyecto.
- Ver un reporte detallado de un diseño de un proyecto.
- El usuario debe poder ingresar todos los datos pertinentes para un proyecto
- El usuario debe poder guardar el estado de un diseño y continuarlo después
- El software debe calcular el diseño con los datos ingresados por el usuario
- El usuario puede revisar el resultado del cálculo para observar la viabilidad del diseño
- El usuario puede ver tanto diseños calculados como diseños sin calcular entre sus diseños creados.

El diseño calculado debe tener una memoria de diseño que pueda ser vista por el usuario

8. DISEÑO

8.1 Interfaz de usuario

La interfaz se diseñó con una estructura clara y lógica, donde los **formularios de entrada de datos** aparecen en la parte superior de la pantalla. Esta disposición permite al usuario revisar fácilmente la información ingresada en cada uno de los **endpoints** relacionados con los distintos aspectos del diseño de pavimentos, como el tránsito, la subrasante, la subbase y la losa de concreto. Cada formulario está optimizado para la **navegación fluida**, con controles de interacción intuitivos que aseguran que los datos se ingresen y se revisan de manera eficiente.

Se utilizó una **paleta de colores sobria** y profesional, evitando elementos visuales innecesarios o distractores que puedan interferir con la concentración del usuario. La tipografía es legible y funcional, reforzando la claridad del contenido. Además, se implementaron **elementos de control bien definidos**, como botones de acción y campos de entrada, que permiten una interacción fluida con el sistema.

La navegación entre las distintas pantallas del sistema es sencilla e intuitiva, permitiendo al usuario moverse entre las diferentes secciones (tránsito, subrasante, subbase, losa de concreto) sin perder de vista los datos ingresados previamente. Cada pantalla ofrece una **vista resumen** de los datos capturados, lo que facilita la revisión y validación de la información antes de realizar los cálculos de diseño.

8.1.1 Funcionalidades Principales

- Crear proyectos
- Visualizar proyectos
- Editar proyectos

8.1.2 Funcionalidades en un proyecto

- Un proyecto contiene varios diseños de varias vías
- A un proyecto se le pueden agregar vías
- Un diseño de un proyecto puede estar calculado o pendiente por ser calculado (esto quiere decir que se ha creado, pero no se ha terminado de hacer el diseño)
- Los diseños quedan guardados en la nube para luego ser vistos desde cualquier otro lado
- Se puede seleccionar una vía dentro del proyecto para tener una vista más detallada

8.1.3 Funcionalidades en un diseño

- El usuario puede visualizar la información del diseño terminado
- El usuario puede editar un diseño terminado o en proceso
- El usuario puede, en el proceso de diseño seleccionar si utilizar uno de los espectros de carga presentes en la base de datos o su propio espectro de carga
- El usuario puede copiar y pegar espectros de carga propios desde Excel para utilizar en el diseño.
- Cada paso del diseño debe tener ayudas en campos requeridos que informen al usuario sobre la parte del diseño que se está llenando.
- El usuario puede, luego de haber ingresado los datos correspondientes a un diseño, dar en un botón calcular diseño y el programa debería dar diseños posibles (la idea de esto es tener la posibilidad de escoger alguno de los diseños que cumplan con lo requerido en el tiempo de vida de un diseño)

8.1.4 Primer Mockup/Prototipo rápido de pantallas

La pantalla principal muestra la información de los proyectos de diseño de pavimentos elaborados por el usuario. En la Figura 11 a la izquierda se muestra la lista de proyectos, en este caso hay un proyecto que tiene dos unidades de diseño (dos estructuras de pavimento). Una vez seleccionado el proyecto, muestra los datos más significativos de un diseño.

Figura 11 Pantalla principal

En la Figura 12 se muestran la pantalla del menú para crear una unidad de diseño nueva.

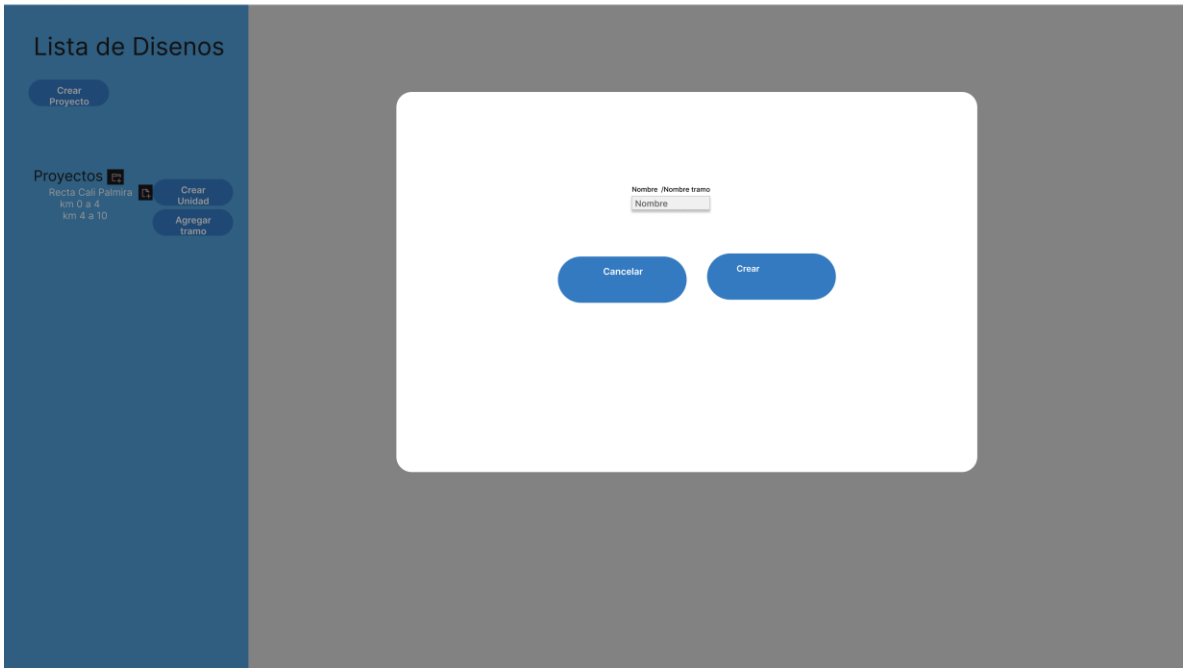


Figura 12. Crear tramo/unidad

En las figuras 13 a 17 se muestra las pantallas de entrada de datos necesarios para el diseño del pavimento. En la parte superior de cada pantalla los botones separan por sección de acuerdo con cada variable de diseño y se muestra en cuál de ellas se está actualmente. Las secciones son las siguientes:

- Volumen de tránsito (Figura 13).
- Clasificación vehicular (Figura 14).
- Espectro de carga (Figura 15).
- Subrasante y subbase (Figura 16).
- Datos del concreto (Figura 17).

En la Figura 18 se muestra la pantalla de resultados del diseño del pavimento para una Unidad de diseño determinada de un proyecto en particular.

Figura 13 Pantalla de volumen de tránsito

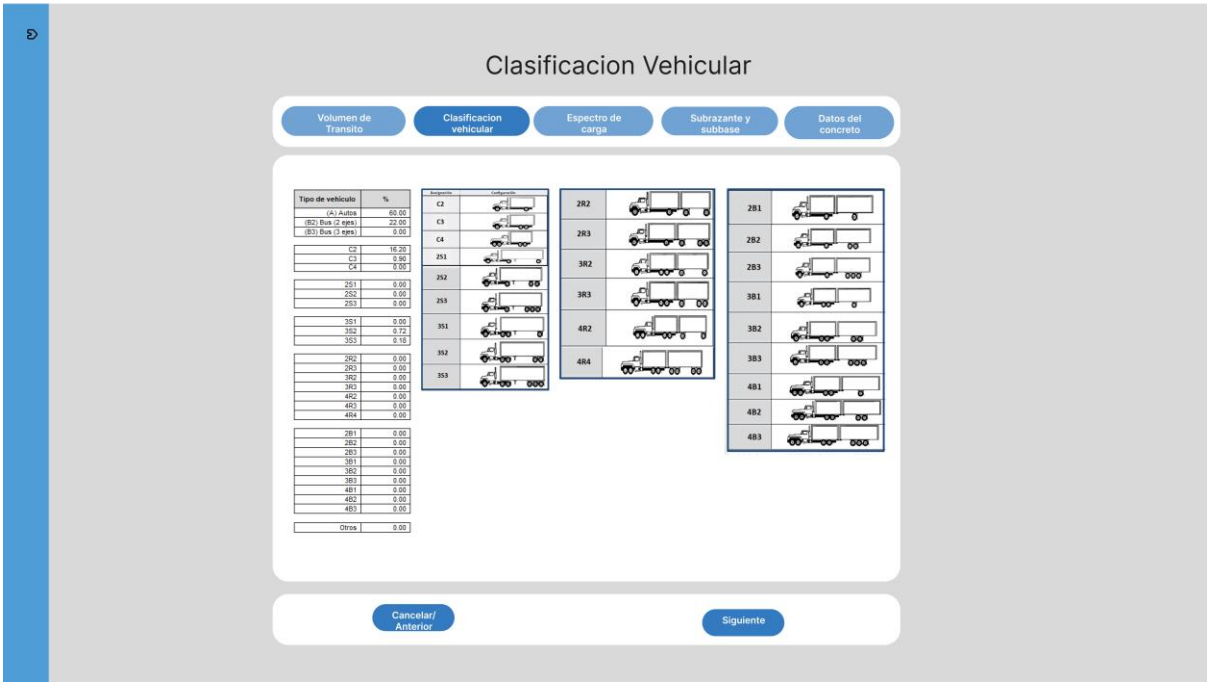


Figura 14 Pantalla clasificación vehicular

Figura 15 Pantalla de espectro de carga

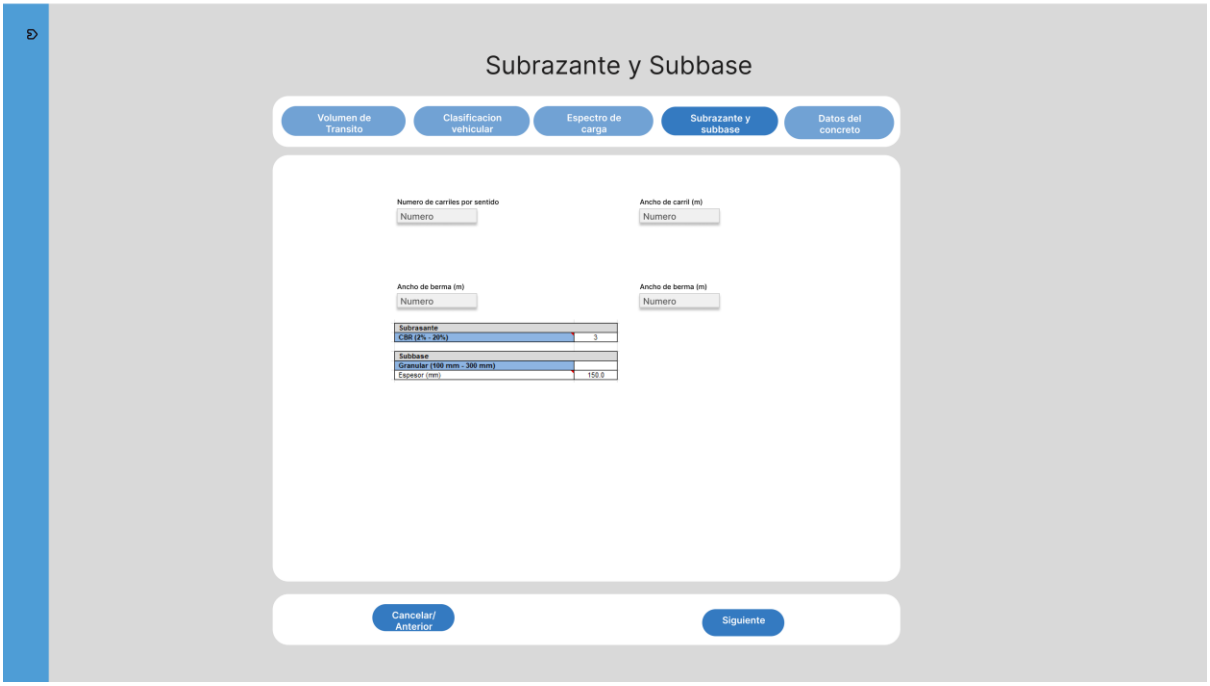


Figura 16 Pantalla de subrasante y subbase

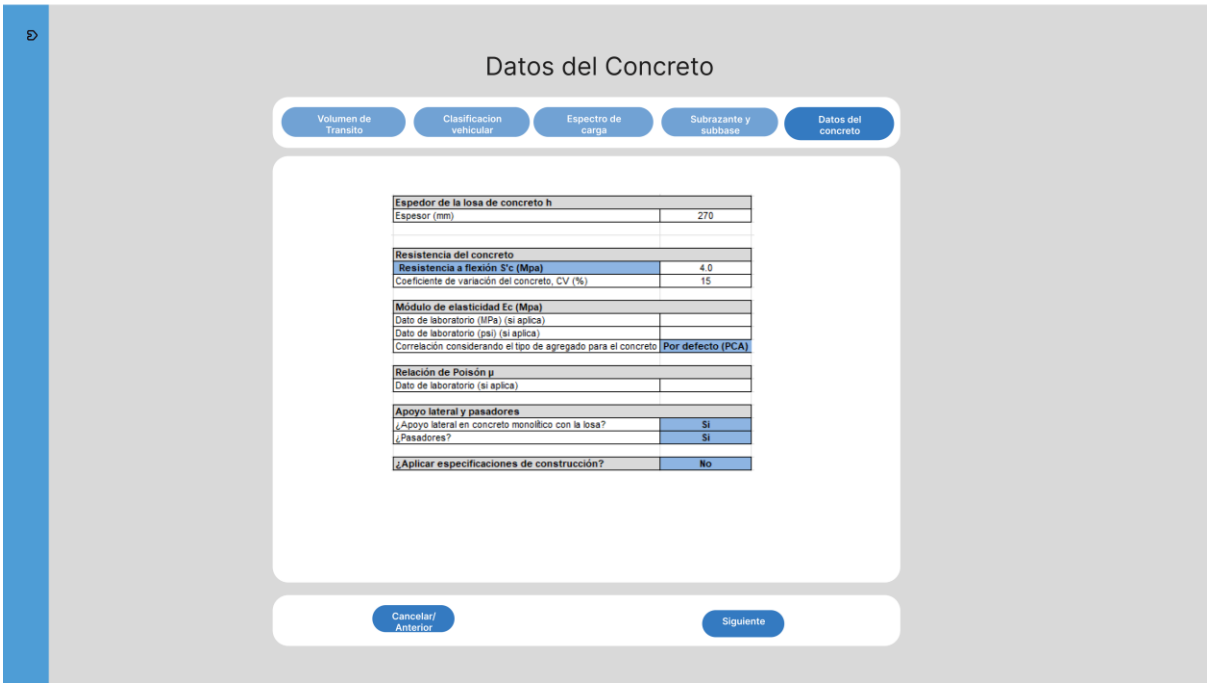


Figura 17 Pantalla de datos del concreto

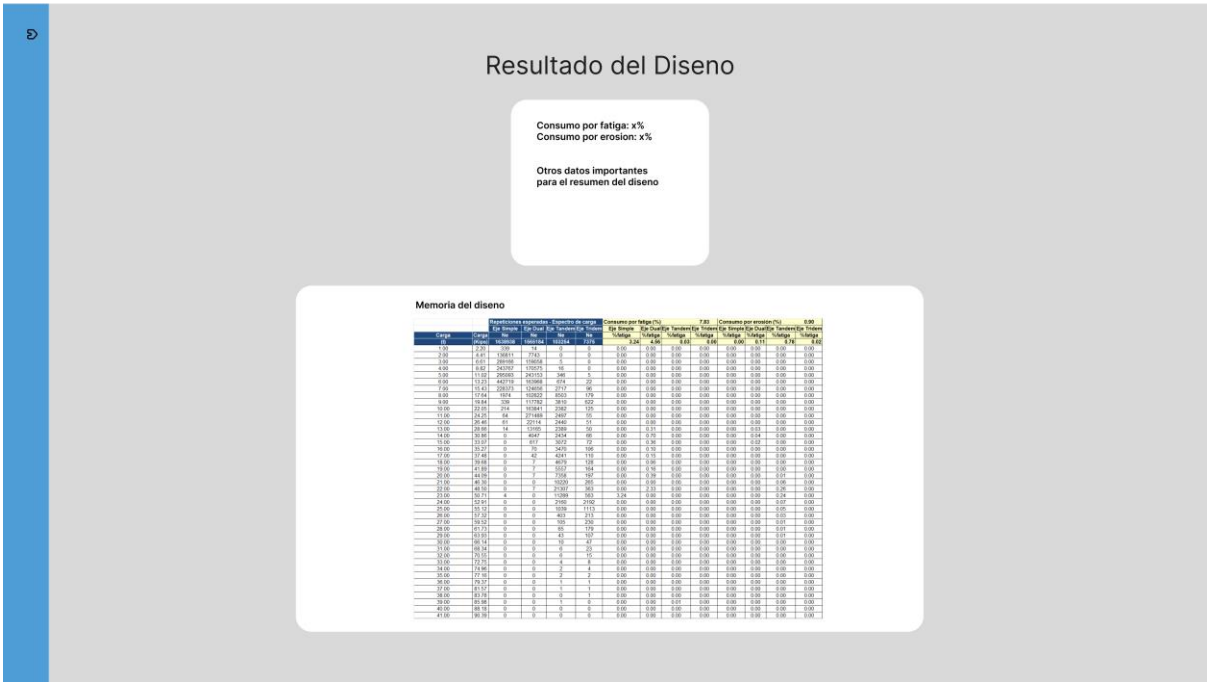


Figura 18 Pantalla de resultados del diseño

En la figura 18 muestra dos secciones. En una se muestra el resultado de un diseño donde se da el consumo por fatiga, consumo por erosión y otros datos relevantes del diseño del pavimento. En la segunda sección se muestra una tabla que especifica la memoria de diseño, la cual muestra de donde provienen los datos.

8.2 Diseño del flujo de la aplicación

En las Figuras 19 y 20 se muestra el diagrama de flujo de la interfaz de usuario y el diagrama de flujo del servicio de backend con los endpoints de la aplicación. Los diagramas se escriben en ingles por costumbre en mi experiencia profesional.

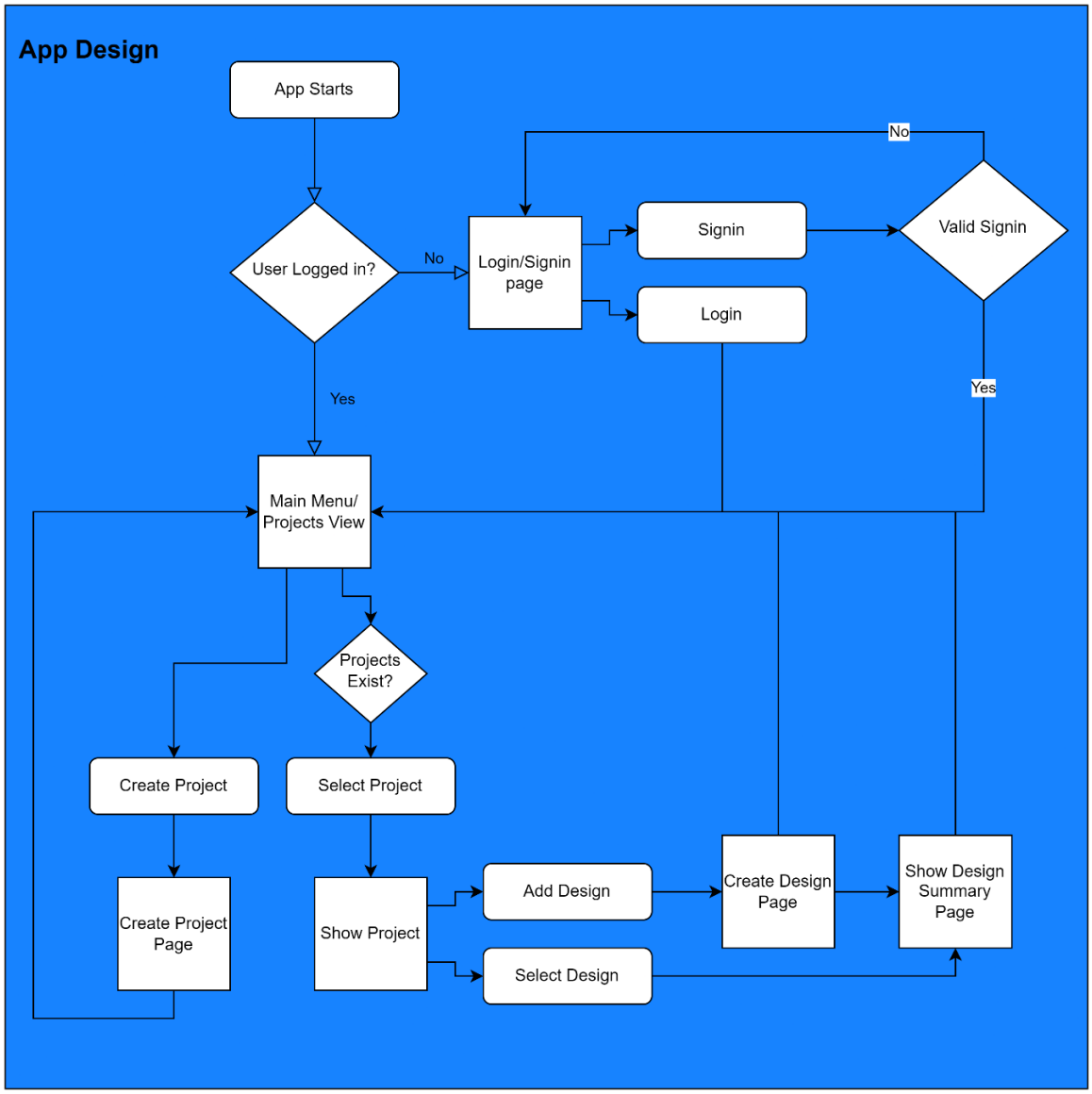


Figura 19. Diagrama de flujo de la interfaz de usuario

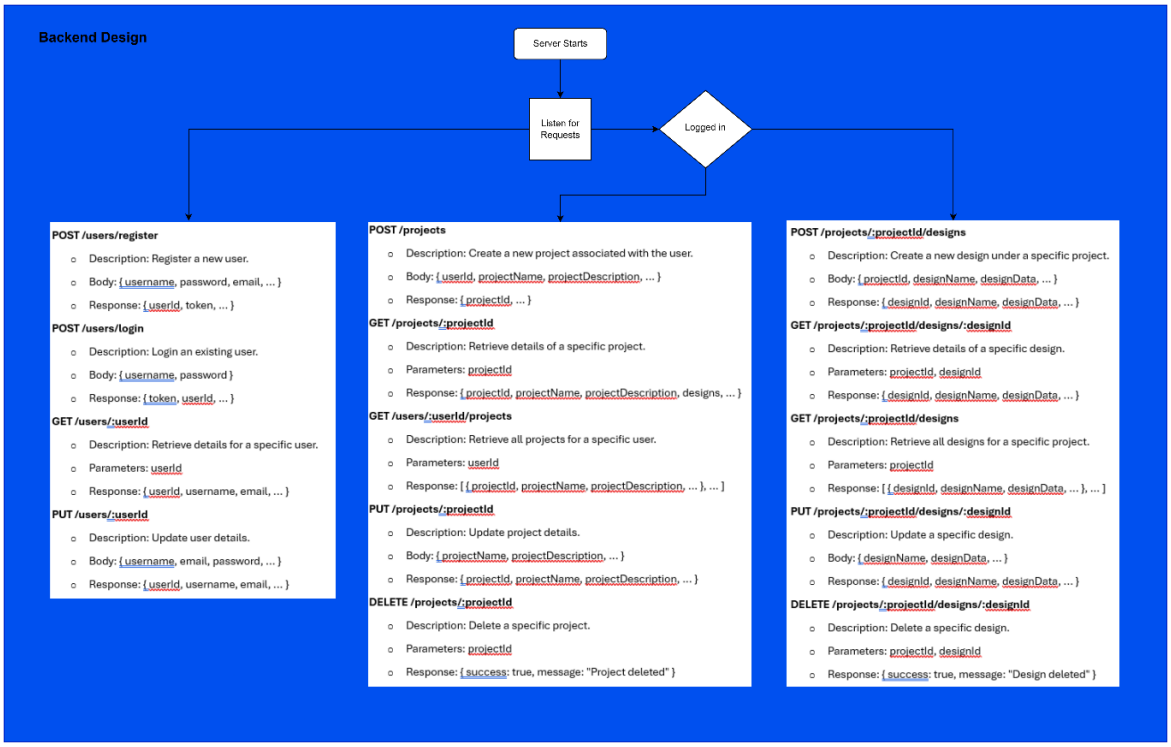


Figura 20. diagrama de flujo del servicio de backend con los endpoints de la aplicación

La figura 20 muestra esencialmente los tres tipos principales de endpoints de la aplicación. En el recuadro de la izquierda se muestran los endpoints relacionados con los usuarios, en el recuadro del medio los endpoints relacionados con los proyectos y en el recuadro de la derecha los endpoints relacionados con los diseños/unidades.

8.3 Diseño de endpoints

Se puede consultar los endpoints de este proyecto de manera más actualizada en el siguiente link: http://35.175.58.164:3000/api-docs/ . En términos generales, los endpoints de usuario realizan operaciones de CRUD para los usuarios, operaciones de autenticación y cierre de sesión en el sistema

8.3.1 Endpoints del usuario

1. POST /users/signup
    - Descripcion: Registrar un nuevo usuario.
    - Body: { username, password, email, ... }
    - Response: { userId }
2. POST /users/login
    - Descripcion: Login un usuario existente.
    - Body: { username, password }
    - Response: { token, userId, ... }
3. GET /users/:userId
    - Descripcion: Obtiene la informacion de un usuario.
    - Parameters: userId
    - Response: { userId, username, email, ... }
4. PUT /users/:userId
    - Description: Update user details.
    - Body: { username, email, password, ... }
    - Response: { userId, username, email, ... }

8.3.2 Endpoints de proyecto

1. POST /projects
    - Description: Crea un Proyecto asociado con un usuario.
    - Authorization: Bearer token
    - Body: { userId, projectName, projectDescription, ... }
    - Response: { projectId, ... }
2. GET /projects/:projectId
    - Description: Obtiene el proyecto, identificado por proyectId, del usuario al que pertenezca la autenticación.
    - Authorization: Bearer token
    - Parameters: projectId
    - Response: { projectId, projectName, projectDescription, designs, ... }
3. PUT /projects/:projectId
    - Description: Actualizar información de un proyecto identificado por projectId.
    - Authorization: Bearer token
    - Body: { projectName, projectDescription, ... }
    - Response: { projectId, projectName, projectDescription, ... }

4. DELETE /projects/:projectId
    - Description: Borra un Proyecto identificado por projectId.
    - Authorization: Bearer token
    - Parameters: projectId
    - Response: { success: true, message: "Project deleted" }

8.3.3 Endpoints de diseño

1. POST /projects/:projectId/designs
    - Description: Crear nuevo diseño asociado a un proyecto.
    - Body: { projectId, designName, designData, ... }
    - Response: { designId, designName, designData, ... }
2. GET /projects/:projectId/designs/:designId
    - Description: Obtiene información de un diseño.
    - Parameters: projectId, designId
    - Response: { designId, designName, designData, ... }
3. GET /projects/:projectId/designs
    - Description: Obtiene todos los diseños de un proyecto.
    - Parameters: projectId
    - Response: [{ designId, designName, designData, ... }, ...]
4. PUT /projects/:projectId/designs/:designId
    - Description: Update a specific design.
    - Body: { designName, designData, ... }
    - Response: { designId, designName, designData, ... }
5. DELETE /projects/:projectId/designs/:designId
    - Description: Borra un diseño identificado por designId del proyecto identificado por projectId.
    - Parameters: projectId, designId
    - Response: { success: true, message: "Design deleted" }

8.3.4 Autenticación/Autorización Endpoints

1. POST /auth/refresh
    - Description: Refresh the user's authentication token.
    - Body: { token }
    - Response: { newToken }

8.3.5 Codigos de respuestas HTTP Utilizados:

- **201**: Success response for user creation.
- **200**: Success response for user login and user detail retrieval.
- **422**: Validation error for the signup route (e.g., user already exists).
- **401**: Unauthorized error for incorrect login credentials.
- **404**: User not found for detail retrieval.
- **500**: General error for server-related issues.

8.4 Modelo de datos

La Figura 21 describe el modelo de datos utilizado. Principalmente se hace énfasis en la relación que existe entre los proyectos que pertenecen a un usuario y los diseños que contiene un proyecto. En este caso en particular, al tratarse de un modelo de datos para una base de datos no relacional, se decide utilizar las relaciones uno a uno para profundizar en la explicación los objetos que están estructurados en el sistema.

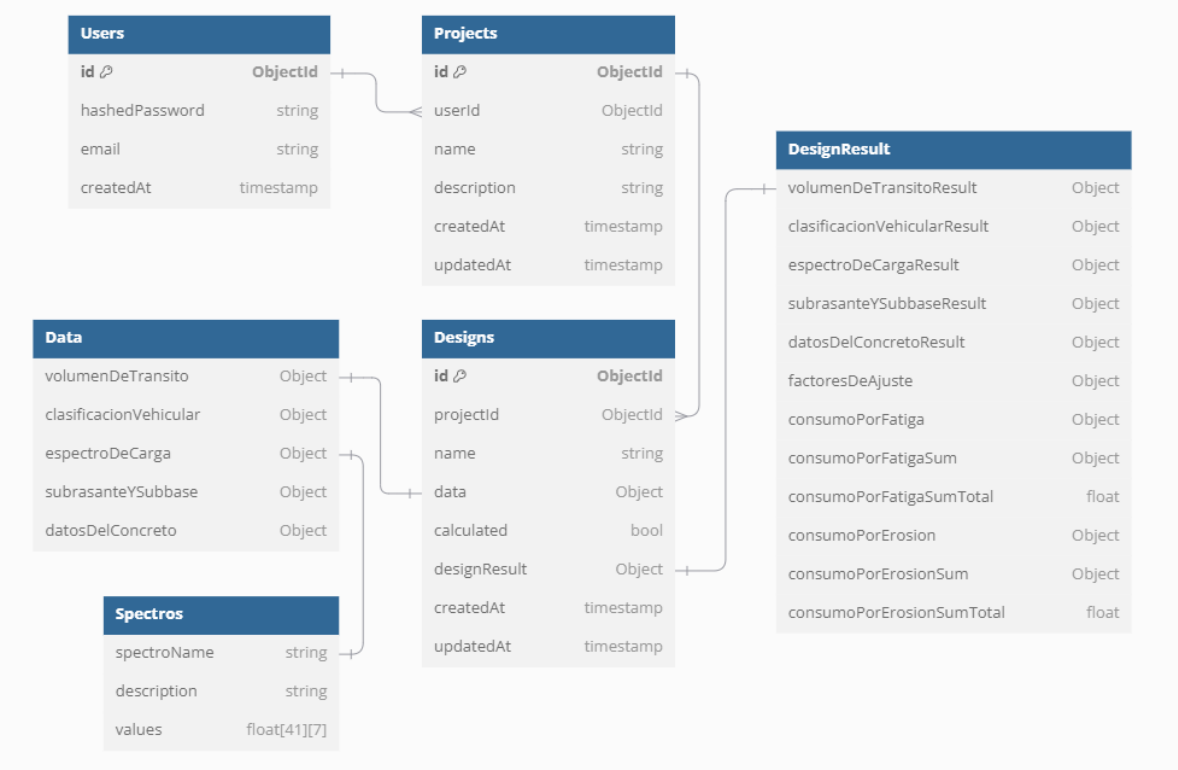


Figura 21. Modelo de datos del sistema

9. IMPLEMENTACIÓN

9.1 Selección de base de datos para el sistema

Una base de datos no relacional ofrece un balance entre flexibilidad, escalabilidad y rendimiento, siendo lo más adecuado para este proyecto debido a que es posible que experimente cambios en futuras versiones. Las bases de datos no relacionales permiten manejar datos no estructurados o semi-estructurados con mayor facilidad, lo que es clave en este proyecto ya que el esquema de los datos podría evolucionar con el tiempo, sin la necesidad de realizar complejas migraciones de esquemas como en las bases de datos relacionales. Otro punto para tener en cuenta al seleccionar una base de datos no relacional es que la complejidad de los queries no es muy alta, lo que hace que las consultas sean eficientes sin la necesidad de un sistema de relaciones entre tablas (Hecht & Jablonski, 2011). Además, la capacidad de escalar horizontalmente, al distribuir los datos en múltiples servidores de forma eficiente, es una ventaja significativa frente a las bases de datos relacionales que requieren un mayor esfuerzo en la escalabilidad vertical. Esto la convierte en una solución más flexible y económica si se prevé un crecimiento en la cantidad de datos o usuarios.

Se escoge MongoDB como motor de base de datos. Esta decisión es guiada por el estado actual del proyecto y los requerimientos cambiantes que se han visto durante su desarrollo y en las reuniones con el experto en pavimentos. Este motor de base de datos permite tener flexibilidad en los esquemas, excelente performance y al ser un proyecto desarrollado en su totalidad por un solo desarrollador, también mejora la velocidad en la que desarrollo en comparación con una base de datos SQL. Hay que recalcar también, que en cuanto a la parte económica del proyecto y para no poner más carga en el servidor donde está montado el backend y frontend del sistema, MongoDB ofrece un plan gratis en línea que es suficiente para el uso estimado de la aplicación.

Como puntos adicionales a tener en cuenta para seleccionar MongoDB como motor de base de datos se tienen:

9.1.1 Esquema flexible

MongoDB tiene una arquitectura sin esquema la cual es perfecta para un entorno cambiante a futuro del proyecto, el cual incluye múltiples usuarios y sus respectivos proyectos asociados. A diferencia de una base de datos relacional, que requiere predefinir esquemas, MongoDB permite que cada documento o iteraciones futuras de los documentos tengan estructuras dinámicas (Chodorow, 2013). Esta flexibilidad le permitirá al proyecto en un futuro agregar módulos nuevos y que se ajusten sin problema alguno mientras el proyecto evoluciona y aumenta su complejidad (Harrison, 2015).

9.1.2 Modelo orientado a documentos

El modelo de documentos de MongoDB almacena los datos como JSON lo cual permite agregar datos extra en los modelos y evitar operaciones costosas como “Join” y mejorar el rendimiento en operaciones de lectura. Por ejemplo, para este caso particular se podría almacenar información de los proyectos de cada usuario junto a datos del usuario siguiendo la jerarquía diseñada anteriormente (Membrey, Plugge, & Hawkins, 2010).

9.1.3 Escalabilidad y rendimiento

MongoDB fue diseñado para escalar horizontalmente distribuyendo los datos a través de múltiples servidores, esto conocido como “sharding” (Schwartz, 2016). Esta funcionalidad asegura un buen rendimiento a medida que tanto el número de usuarios como la complejidad del proyecto crecen, al simplemente agregar más servidores de bases de datos (Han & Strickland, 2016).

9.1.4 Requerimientos futuros

La base de datos no relacional como MongoDB permite que el proyecto evolucione para incluir funcionalidades nuevas o realizar cambios.

9.2 Selección de tecnologías para el Backend

Esta decisión se basa en factores como rendimiento, escalabilidad, eficiencia del desarrollo y la perfecta integración con la tecnología frontend elegida, React. Cabe recalcar que, al ser un proyecto realizado en su totalidad por un solo desarrollador, es una buena opción utilizar el mismo lenguaje tanto para el Frontend como para los servicios del Backend facilitando y optimizando la velocidad del desarrollo del proyecto.

9.2.1 Rendimiento y escalabilidad de NodeJS

Node.js se basa en el motor JavaScript V8, conocido por su alto rendimiento. La arquitectura sin bloqueos y basada en eventos de Node.js permite gestionar una gran cantidad de conexiones simultáneas de manera eficiente (Tilkov & Vinoski, 2010). Esto es bueno para una aplicación web con múltiples usuarios que interactúan con el sistema en tiempo real, como en una plataforma de intercambio de diseños.

Por otra parte, las operaciones de E/S asincrónicas de Node.js permiten gestionar múltiples solicitudes sin esperar a que se complete ninguna. Esta naturaleza no bloqueante garantiza que el servidor pueda gestionar altos niveles de concurrencia con un consumo mínimo de recursos (Osei, 2016).

Node.js uso de JavaScript lo cual permite un desarrollo rápido y eficiente (Cantelon, Harter, Holowaychuk, & Rajlich, 2018). Esta uniformidad permite escribir código tanto del lado del cliente como del lado del servidor en el mismo lenguaje lo aumenta la productividad. El uso de JavaScript tanto para el frontend (React) como para el backend (Node.js) agiliza el proceso de desarrollo ya que elimina la necesidad de cambiar de contexto entre distintos lenguajes de programación y facilita una base de código más fácil de mantener (Holmes, 2012).

9.2.2 Integración con React

React es una biblioteca de interfaz de usuario que se combina con Node.js para crear una aplicación JavaScript. La compatibilidad entre React y Node.js simplifica el flujo de trabajo y facilita la gestión del estado en toda la aplicación.

Server-Side Rendering (SSR): Node.js es ideal para la representación del lado del servidor de aplicaciones React, lo que puede mejorar el rendimiento y el SEO (Seo & Jung, 2020). Esta capacidad es beneficiosa para aplicaciones que requieren tiempos de carga iniciales rápidos y una mejor visibilidad en los motores de búsqueda. Para el caso de este proyecto en particular no entra en los objetivos, pero utilizar estas tecnologías permite migrar fácilmente a este tipo de esquema si una mejora en el rendimiento es necesaria.

Integración de API: Node.js se destaca en la creación de API que pueden ser utilizadas por aplicaciones React. Las API RESTful o los puntos finales GraphQL creados con Node.js se pueden integrar fácilmente con el frontend de React, lo que garantiza un flujo de datos fluido y una interacción entre el cliente y el servidor (Wexler, 2017).

9.2.3 Ecosistema sólido y apoyo comunitario

Node.js tiene un ecosistema dinámico y extenso, con una gran cantidad de bibliotecas y marcos disponibles a través de npm (Node Package Manager). Este ecosistema acelera el desarrollo al proporcionar módulos prediseñados para tareas comunes (Casciaro & Mammino, 2020).

Microservicios y arquitectura modular: la naturaleza modular de Node.js fomenta el desarrollo de microservicios, lo que permite que las distintas partes de la aplicación se desarrollen y escalen de forma independiente. Esta arquitectura se alinea bien con la naturaleza evolutiva del sistema de gestión de diseño de usuarios, donde se pueden agregar nuevas funciones de forma incremental (Jangda, Qadeer, Waseem, & Karim, 2018).

9.2.4 Experiencia y familiaridad del desarrollador

Se eligió Node.js como la tecnología de backend para el sistema debido a su rendimiento, escalabilidad e integración perfecta con React. Su arquitectura sin

bloqueos y basada en eventos garantiza un manejo eficiente de solicitudes simultáneas, mientras que la uniformidad del uso de JavaScript tanto en el backend como en el frontend mejora la velocidad y la eficiencia del desarrollo (Kappagantula & Kalvakuntla, 2016). La experiencia previa existente en Node.js y React, solidifica aún más esta elección, asegurando una solución consistente y mantenible para las necesidades actuales y futuras.

Una de las principales razones para elegir Node.js es la experiencia previa y la familiaridad con esta tecnología. Tener experiencia previa con Node.js y React reduce significativamente la curva de aprendizaje y el tiempo de desarrollo, lo que permite centrarse en la creación de las características y la funcionalidad de la aplicación.

9.3 Selección de tecnología para el frontend del sistema

Esta elección está respaldada por su rendimiento, escalabilidad, arquitectura basada en componentes e integración con Node.js en el backend. Además, la uniformidad de usar JavaScript tanto para el frontend como para el backend simplifica el desarrollo, lo cual es crucial dado que un solo desarrollador maneja todo el proyecto.

9.3.1 Razones para elegir React

- **Arquitectura basada en Componentes**: La arquitectura basada en componentes de React permite la creación de componentes de interfaz de usuario reutilizables y autónomos. Este modularidad facilita el desarrollo de interfaces de usuario complejas al dividirlas en partes más pequeñas y manejables (Banks, Porcello, & Porcello, 2020).

- **Reusabilidad:** Los componentes se pueden reutilizar en distintas partes de la aplicación, lo que promueve la coherencia y reduce el tiempo de desarrollo. Esto resulta especialmente útil en una plataforma de intercambio de diseños en la que elementos como los perfiles de usuario y las galerías de diseño se pueden estandarizar y reutilizar (Gackenheimer, 2015).

- **Mantenibilidad:** La separación de preocupaciones inherente a la arquitectura de React hace que la base de código sea más fácil de mantener y ampliar. Cada componente maneja su propia lógica y presentación, lo que simplifica la depuración y la incorporación de funciones futuras (Bevacqua, 2018).

- **Rendimiento y Virtual DOM**: React utiliza un DOM virtual para gestionar y actualizar de forma eficiente la interfaz de usuario. El DOM virtual es una

representación liviana del DOM real, lo que permite a React realizar actualizaciones mínimas y reducir la sobrecarga asociada con la manipulación directa del DOM (Griffith, 2018).

- **Actualizaciones eficientes:** El algoritmo de reconciliación de React garantiza que solo se actualicen las partes del DOM que han cambiado, lo que genera una representación más rápida y un mejor rendimiento. Esto es fundamental para una experiencia de usuario responsiva, especialmente en una aplicación web con contenido dinámico como diseños de usuario (Makin, 2017).

- **Integración con Node.js**: React se combina con Node.js, lo que facilita un buen flujo de trabajo de desarrollo y una representación eficiente del lado del servidor (SSR) cuando es necesario. (Seo & Jung, 2020). Esta integración es beneficiosa para el rendimiento general y el SEO de la aplicación.

- **Flujo de datos consistente:** El uso de React para el frontend y Node.js para el backend permite un flujo de datos y una gestión de estados consistentes en toda la aplicación. Esta coherencia garantiza que la lógica de la aplicación se mantenga unificada, lo que simplifica el desarrollo y el mantenimiento (Cantelon, Harter, Holowaychuk, & Rajlich, 2018).

9.3.2 Eficiencia en el Desarrollo utilizando javascript como lenguaje

La elección de React para el frontend complementa el uso de Node.js en el backend, lo que permite un entorno de desarrollo de JavaScript completo. Esta uniformidad simplifica el proceso de desarrollo, en particular para un desarrollador individual.

- **Full-Stack Development:** Escribir código tanto del lado del cliente como del lado del servidor en JavaScript reduce la carga cognitiva y el cambio de contexto entre diferentes lenguajes, lo que genera una mayor productividad y una base de código más cohesiva (Holmes, 2012).

- **Experiencia del desarrollador:** La familiaridad con JavaScript y la experiencia con React contribuyen a un proceso de desarrollo más rápido y eficiente (Kappagantula & Kalvakuntla, 2016). Aprovechar las habilidades existentes permite centrarse en implementar características y funcionalidades en lugar de aprender nuevos lenguajes y marcos de trabajo.

9.3.3 Comunidad y soporte robustos

React cuenta con un ecosistema sólido con una gran cantidad de bibliotecas, herramientas y recursos comunitarios que facilitan el desarrollo. Esta red de soporte mejora la productividad y brinda soluciones a los desafíos de desarrollo comunes.

- **Librerías de terceros:** El ecosistema de React incluye bibliotecas y herramientas de terceros que amplían su funcionalidad. Estas bibliotecas ayudan a acelerar el desarrollo al proporcionar soluciones predefinidas para tareas comunes, como la gestión de estados, el enrutamiento y el manejo de formularios (Casciaro & Mammino, 2020).
- **Soporte de la comunidad:** Una comunidad grande y activa significa que hay muchos recursos para aprender y solucionar problemas. Hay tutoriales, foros y documentación disponibles, lo que facilita superar obstáculos y mantenerse actualizado con las mejores prácticas (Banks, Porcello, & Porcello, 2020).

9.3.4 Comparación con otros Frameworks Frameworks

- Angular

Angular es un marco integral que proporciona una estructura sólida para crear aplicaciones. Sin embargo, su pronunciada curva de aprendizaje y su complejidad pueden no ser ideales para un desarrollador independiente que busque una solución más sencilla (Wasson, 2017).

Complejidad**:** El amplio conjunto de características y la estructura estricta de Angular pueden resultar abrumadores, especialmente para proyectos en los que se requiere un desarrollo rápido y flexibilidad (Porebski, Godfrey, & Wijngaarde, 2011).

- Vue.js

Vue.js es otro framework de interfaz popular que se caracteriza por su simplicidad y facilidad de integración. Sin embargo, la comunidad más grande de React y su ecosistema más amplio ofrecen más recursos e integraciones de terceros, que son cruciales para un desarrollador independiente (Kniazev, 2018).

Ecosistema**:** Si bien Vue.js es flexible y fácil de aprender, el ecosistema de React proporciona herramientas y bibliotecas más completas, que pueden acelerar el desarrollo (Casciaro & Mammino, 2020).

9.4 Decisiones de arquitectura en el Backend del proyecto

Se siguen puntos clave sobre principios de diseño como separación de responsabilidades, modularidad y escalabilidad. Las decisiones hechas durante la fase de diseño permiten manejar los requerimientos del sistema, así como también tener un proyecto con una buena base para un futuro crecimiento de la aplicación. Con una clara separación entre enrutamiento, servicios y lógica de la base de datos, el sistema se puede mantener fácilmente, extender y escalar apropiadamente según futuros cambios en demanda de usuarios o lógica del negocio.

9.4.1 Separación de responsabilidades

Un principio fundamental que guía la arquitectura de este backend es la separación de tareas. El backend está dividido en capas y módulos, cada uno responsable de una función específica. Esto garantiza que cada componente solo se ocupe de una única tarea, lo que hace que el sistema sea más fácil de entender, mantener y ampliar.

- Capa de rutas: las rutas son responsables de manejar las solicitudes y respuestas HTTP. Cada ruta se asigna a un punto final específico y delega la lógica empresarial real a la capa de servicio correspondiente. Este diseño permite que la lógica de enrutamiento se centre únicamente en procesar y enrutar las solicitudes HTTP.

- Capa de servicio: la capa de servicio encapsula la lógica empresarial. Actúa como intermediario entre las rutas y la base de datos, lo que garantiza que las operaciones como la creación, actualización y eliminación de entidades se realicen de acuerdo con las reglas empresariales. Al aislar esta lógica de las rutas, garantizamos que la funcionalidad principal del backend no esté vinculada a las especificidades de HTTP, lo que permite cambios futuros en las capas de transporte (como cambiar a WebSockets) con una refactorización mínima.

- Capa de base de datos: las interacciones con la base de datos se abstraen en una capa separada, db.js. Esta capa maneja la comunicación directa con la base de datos, incluida la consulta y actualización de datos. Al aislar la lógica de la base de datos en su propio módulo, desacoplamos la lógica empresarial de la implementación de la base de datos. Esto hace que sea fácil cambiar a un motor de base de datos diferente en el futuro, como migrar de MongoDB a PostgreSQL, sin cambiar la lógica comercial principal.

9.4.2 Patrones de diseño

Patrón de arquitectura por capas: se implementa un patrón de arquitectura en capas, lo cual permite una clara separación de responsabilidades y facilita la mantenibilidad y escalabilidad del sistema. Esta arquitectura consta de varias capas, como la capa de rutas, la capa de servicios, y la capa de acceso a datos. La capa de rutas se encarga de recibir las solicitudes HTTP y pasar la información necesaria a

la capa de servicios, donde se implementa la lógica de negocio. Finalmente, la capa de acceso a datos se encarga de interactuar con la base de datos, garantizando que las operaciones de lectura y escritura estén desacopladas de las demás capas. Esta estructura modular facilita la modificación y prueba de cada parte del sistema sin afectar el resto del código.

Middleware para autenticación y manejo de errores: Las funciones de middleware, como el middleware de autenticación, garantizan que las cuestiones transversales, como la seguridad, se gestionen de manera centralizada. La lógica de autenticación, que comprueba la validez de los tokens JWT, se aplica globalmente en todas las rutas, lo que garantiza que no se pueda acceder a los puntos finales protegidos sin credenciales válidas.

Además, el middleware de gestión de errores personalizado centraliza la forma en que se gestionan los errores en toda la aplicación. Cuando se produce un error en cualquier ruta o servicio, se transmite a través del middleware de gestión de errores, que genera las respuestas HTTP adecuadas. Esto facilita la ampliación del sistema de gestión de errores en el futuro, añadiendo tipos específicos de registro o notificaciones si es necesario.

9.4.3 Escalabilidad

La arquitectura del backend se ha diseñado teniendo en mente la escalabilidad futura, tanto en términos de crecimiento de la base de código como de posibles optimizaciones del rendimiento.

Arquitectura modular: Al mantener la lógica modular, garantizamos que se puedan agregar nuevas funciones al sistema sin afectar la funcionalidad existente. Por ejemplo, si una nueva función requiere agregar más lógica de negocios, se puede crear un nuevo servicio con sus propias rutas, lo que garantiza que los cambios no interfieran con otras partes del sistema.

Independencia de la base de datos: La capa de base de datos se abstrae de modo que el sistema no esté estrechamente acoplado a una tecnología de base de datos específica. Actualmente, se utiliza MongoDB, pero esta abstracción permite migrar fácilmente el backend a una base de datos relacional como PostgreSQL o MySQL en el futuro sin tener que realizar grandes modificaciones en la capa de servicio o la lógica empresarial.

Manejo eficiente de datos: Para garantizar la escalabilidad en términos de manejo de datos, el backend utiliza las mejores prácticas para administrar grandes conjuntos de datos, como la paginación al obtener recursos y el envío de los datos necesarios únicamente en las respuestas. Esto minimiza el tamaño de la carga útil

para los clientes, lo que mejora la capacidad del sistema para manejar una gran cantidad de solicitudes simultáneas sin degradación del rendimiento.

9.4.4 Autenticación y autorización

La autenticación y la autorización se gestionan mediante tokens web JSON (JWT), lo cual sirve para proteger las API RESTful. El enfoque basado en tokens garantiza que los usuarios se autentiquen con cada solicitud y permite que el backend se escale fácilmente en un entorno distribuido (por ejemplo, una arquitectura de microservicios), donde el almacenamiento de sesiones centralizado podría convertirse en un cuello de botella.

Además, el control de acceso basado en roles (RBAC) se puede agregar fácilmente a este sistema ampliando el sistema de gestión de usuarios para manejar roles y permisos, haciéndolo escalable para aplicaciones que requieren mecanismos de control de acceso complejos.

9.4.5 Manejo y registro de errores

El backend utiliza un sistema de gestión de errores unificado. Se implementó una clase de error personalizada (AppError) para centralizar la gestión de errores en todos los servicios. Esto garantiza respuestas de error consistentes y reduce la duplicación en todo el código base.

Los errores se clasifican según su tipo (por ejemplo, errores de validación, errores de base de datos, errores de autenticación) y cada tipo de error se asigna al código de estado HTTP correspondiente. Esto no solo garantiza la claridad para el cliente, sino que también simplifica la depuración durante el desarrollo y la producción.

9.4.6 Testing y Mocking

El backend se diseñó teniendo en cuenta la capacidad de prueba. Se crearon pruebas unitarias para los servicios, las rutas y la capa de base de datos para garantizar que cada parte del sistema funcione de forma independiente. Por ejemplo, las operaciones de la base de datos se simulan durante las pruebas, lo que garantiza que las pruebas se ejecuten de forma aislada de la base de datos real, lo que acelera el proceso de prueba y evita efectos secundarios.

La simulación también facilita la refactorización o el intercambio de componentes sin interrumpir las pruebas, lo que garantiza que los cambios sean

aislados y menos riesgosos. Además, al separar las responsabilidades correctamente en el sistema se facilita realizar pruebas más puntuales probando los servicios de manera aislada.

9.4.7 A prueba de futuro y extensibilidad

Se han tomado varias decisiones de diseño para garantizar que el backend pueda responder correctamente a cambios o necesidades futuras.

Escalabilidad de la base de datos: A medida que aumentan los datos, se pueden implementar estrategias de fragmentación o replicación en la capa de base de datos de MongoDB para garantizar una alta disponibilidad y tolerancia a fallas. La arquitectura desacoplada también permite cambiar a una base de datos diferente o agregar compatibilidad con múltiples bases de datos con un impacto mínimo en la lógica de la aplicación.

Lista para microservicios: El diseño del backend puede evolucionar hacia una arquitectura de microservicios, donde cada dominio (por ejemplo, usuarios, proyectos, diseños) puede separarse como su propio servicio, comunicándose a través de un intermediario de mensajes o una REST API. Esto se hace más fácil gracias a la separación de preocupaciones, donde cada servicio ya encapsula una parte específica de la lógica empresarial.

Integración con el Frontend: El diseño claro y RESTful de la API garantiza que el frontend pueda interactuar fácilmente con el backend. Los puntos finales de la API siguen las convenciones RESTful, lo que hace que sea intuitivo para los desarrolladores frontend comprender y usar el backend. Además, a medida que la aplicación crece, se pueden introducir funciones como GraphQL para permitir una consulta de datos más eficiente. También, se ha documentado cada endpoint utilizando Swagger el cual ayuda a mostrar documentación detallada de cómo se utiliza, los requerimientos y las posibles respuestas de cada endpoint desarrollado.

9.5 Resultado de la aplicación

A continuación (Figuras 22 a 32), se muestran pantallazos de la aplicación completada para poder visualizar la comparación entre el primer mockup y el resultado final, evidenciando la evolución que tuvo el proyecto en su proceso desde un inicio hasta su fin.

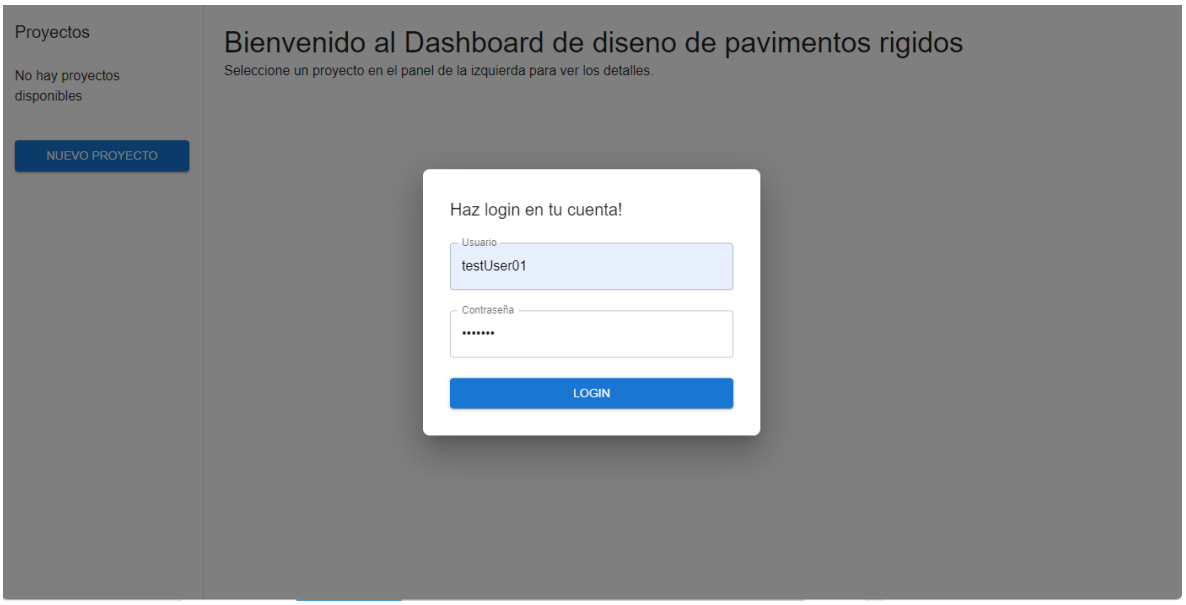


Figura 22 Pantalla de login

Home

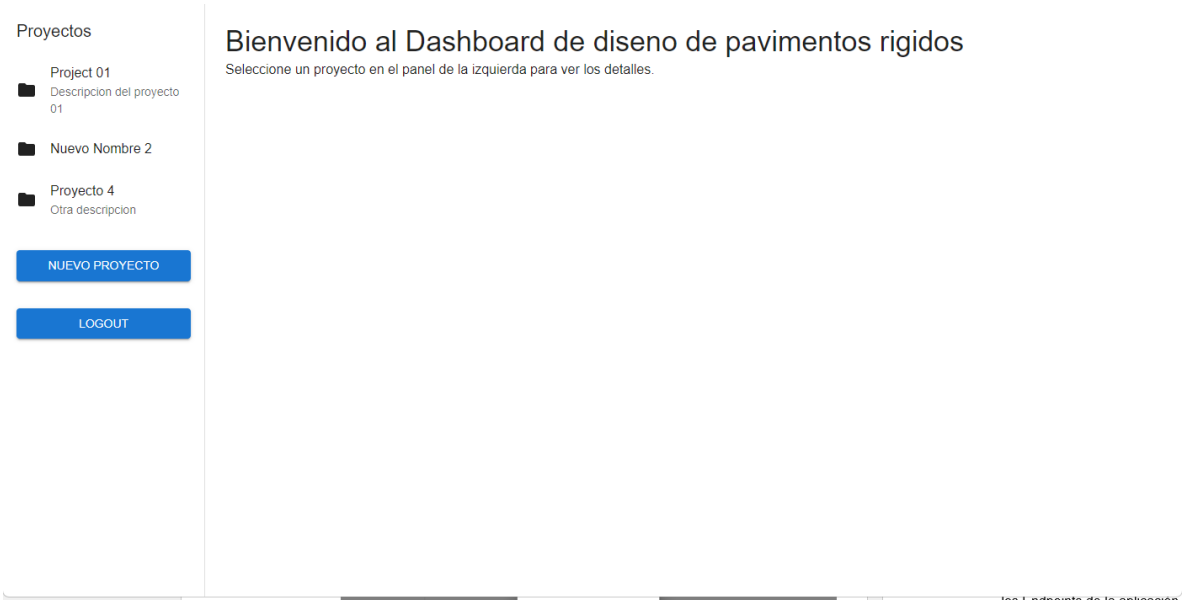


Figura 23. Pantalla de creación de proyecto

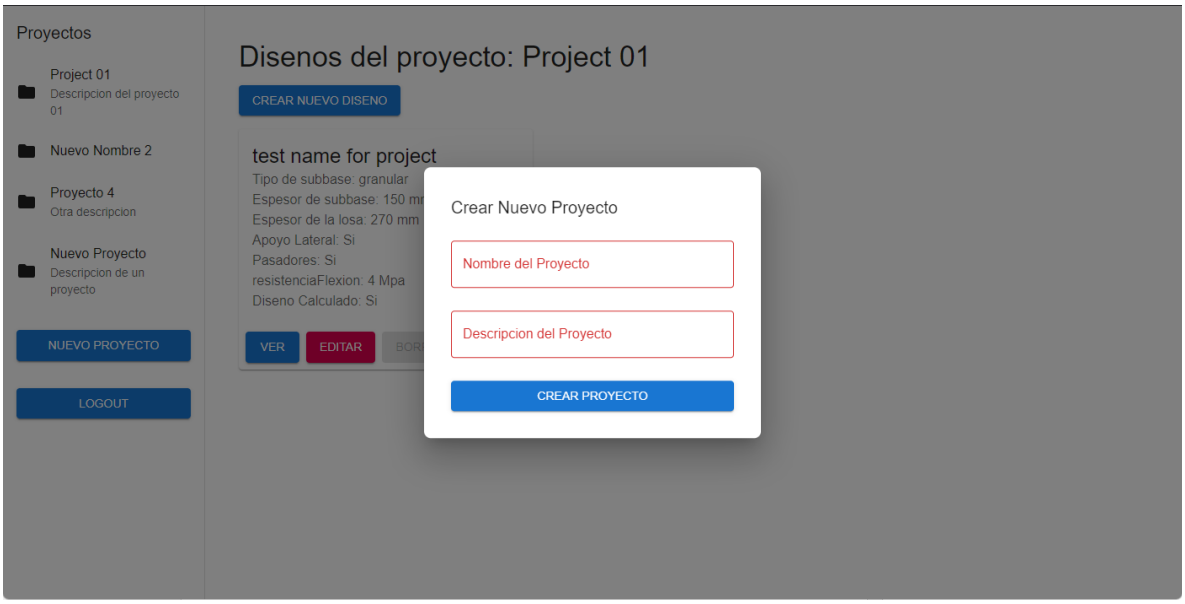


Figura 24. Pantalla de creación de nuevo diseño

Figura 25. Vista de diseños en un proyecto

Figura 26. Entrada da datos necesarios para calcular un diseño

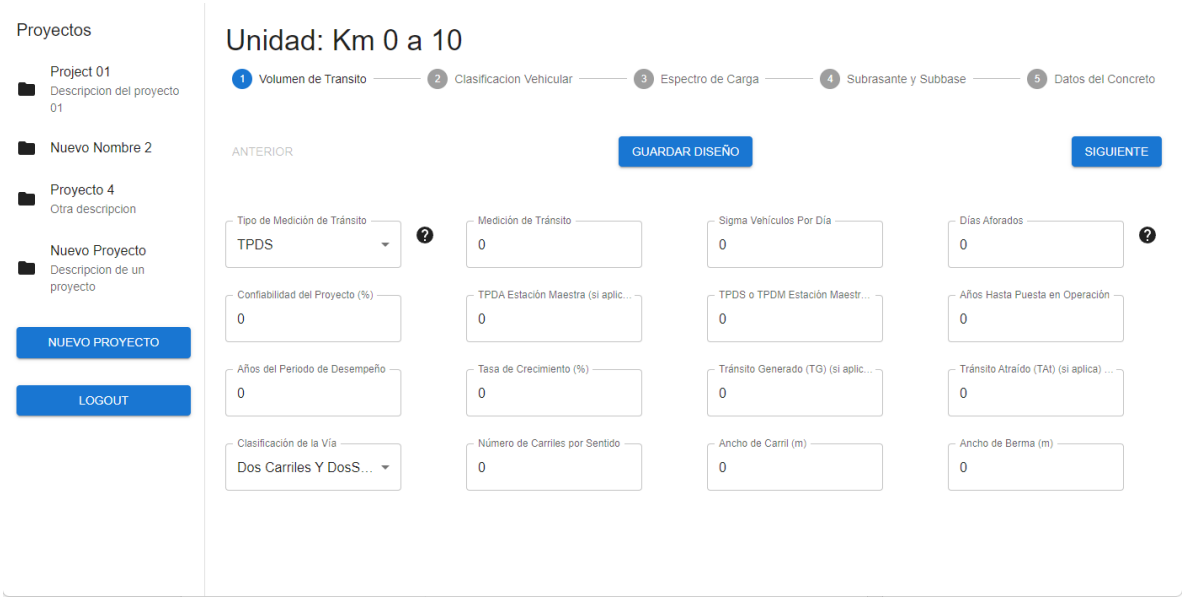


Figura 27. Entrada da datos de volumen de tránsito

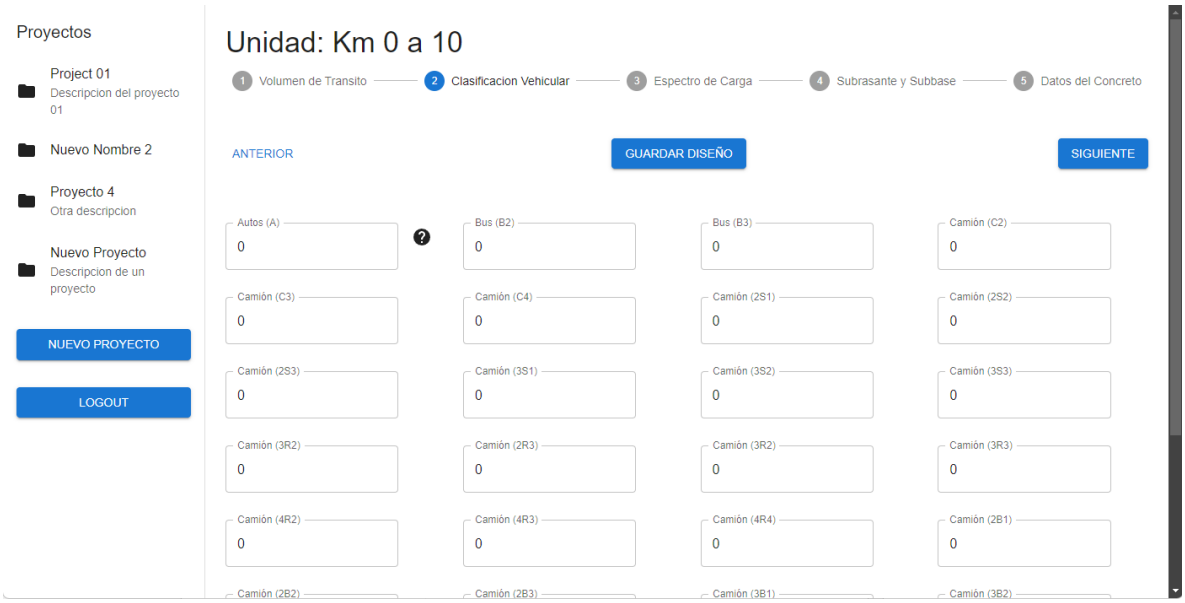


Figura 28. Entrada da datos de clasificación vehicular

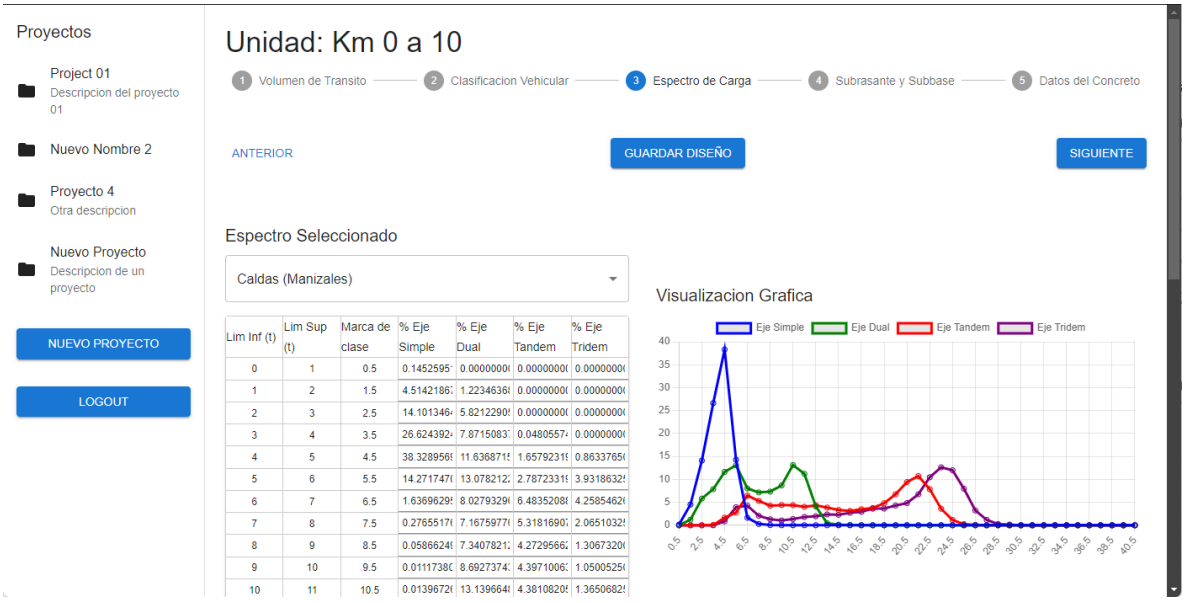


Figura 29. Entrada da datos de espectro de carga

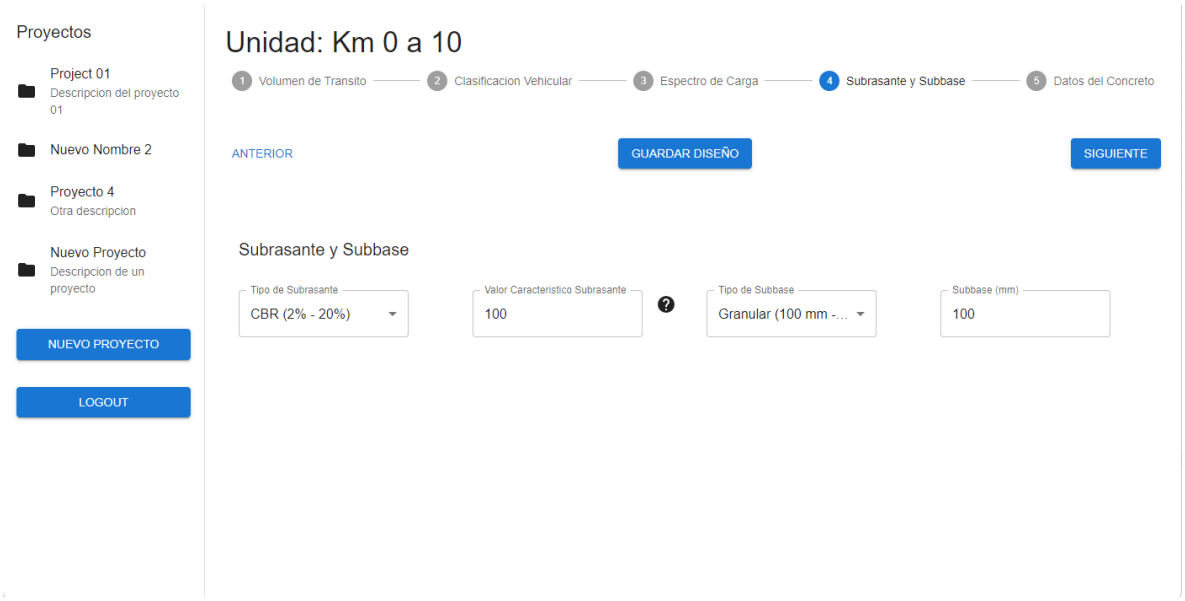


Figura 30. Entrada da datos de subrasante y subbase

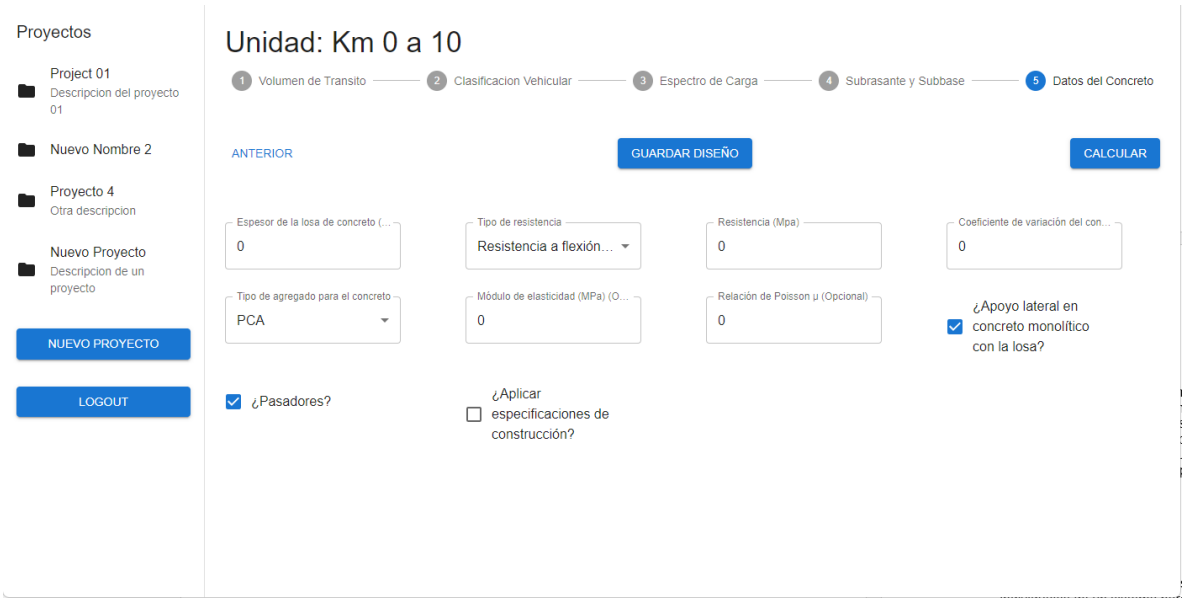


Figura 31. Entrada da datos del concreto

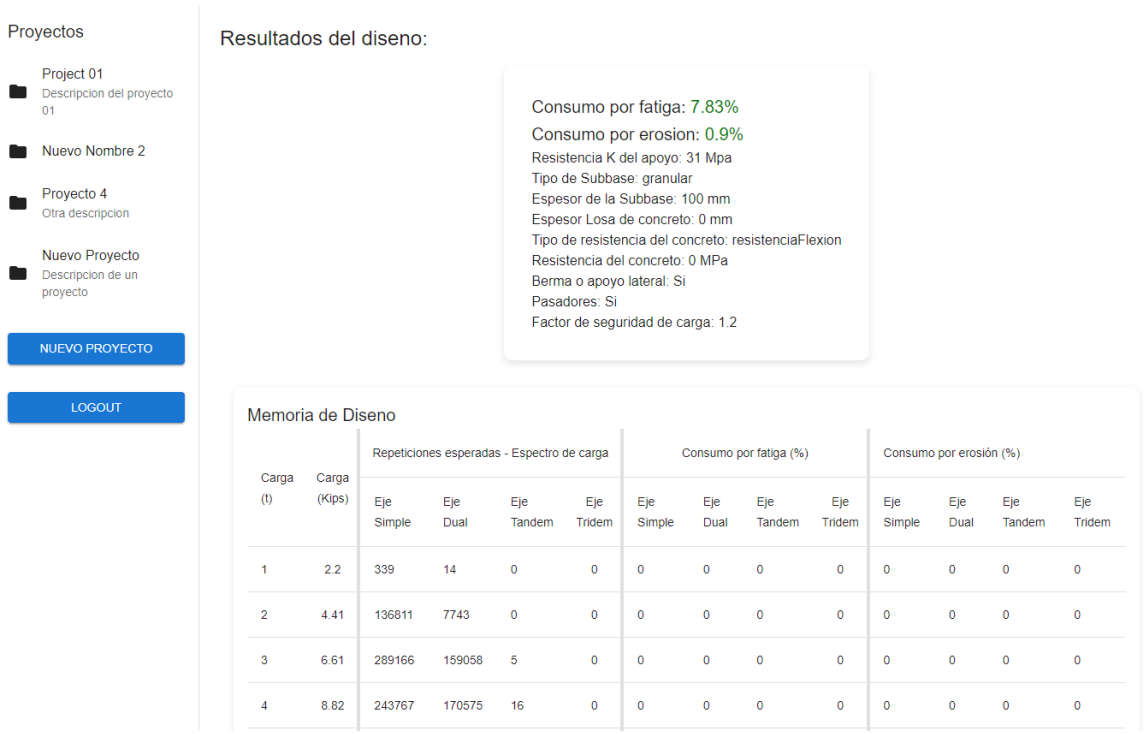


Figura 32. Pantalla donde se muestra el resultado del diseño

Basado en el diseño que se presentó como primer mockup en el capítulo 7 de diseño, se logra una evolución en la interfaz gráfica de la aplicación luego de varias iteraciones con el cliente. El diseño de interfaz de la aplicación, como se ingresan los datos y cuales datos se muestran fueron relevantes para el rediseño grafico de las pantallas de entrada de datos, la pantalla final de resultado de diseño (Figura 32) y la pantalla donde se muestra un breve resumen de los diseños en un proyecto (Figura 26).

Además, la pantalla mostrada en la Figura 29 donde se introducen los datos de un espectro de carga, contiene la base de datos de los espectros de carga de todas las regionales colombianas y permite copiar y pegar datos directamente desde Excel los cuales eran requerimientos importantes del cliente.

10. PRUEBAS

Las pruebas son un componente fundamental en el desarrollo de software, ya que garantizan que el sistema se comporte como se espera y cumpla con sus requisitos. En este proyecto, se utilizó un enfoque de pruebas integral que abarcó varias capas de pruebas, incluidas pruebas unitarias, pruebas de integración y pruebas manuales. Este capítulo describe las metodologías, herramientas y técnicas utilizadas para probar los componentes frontend y backend de este proyecto.

10.1 Pruebas unitarias con Jest y SuperTest

Las pruebas unitarias son el proceso de testear unidades o componentes individuales de un sistema para verificar que cada unidad funciona correctamente de forma aislada. En este proyecto, se utilizó Jest junto con SuperTest, un marco de pruebas de JavaScript, para escribir pruebas unitarias para los servicios y rutas del backend. Las pruebas unitarias se dividieron en dos capas, la lógica de la capa de servicio, lo que garantiza que las reglas de negocio y las manipulaciones de datos fueran correctas y la capa de enrutamiento donde se prueban individualmente que los Endpoints de la aplicación respondan correctamente a escenarios diferentes.

10.1.1 Estrategia de pruebas unitarias

Los servicios de backend se probaron simulando dependencias, en particular la base de datos (db.js) y otros servicios, para garantizar que las pruebas estuvieran aisladas y centradas en la lógica que se estaba probando. Las simulaciones se crearon utilizando las funciones de simulación de Jest (jest.fn() y jest.mock()), lo que permitió simular el comportamiento de la base de datos sin acceso real a los datos.

10.1.2 Ejemplo de prueba unitaria

Como la aplicación fue desarrollada siguiendo una arquitectura por capas, es necesario realizar pruebas unitarias tanto en la capa de servicio como en la capa de enrutamiento/control. A continuación, se da un ejemplo de prueba unitaria en la capa de servicio seguido por una prueba unitaria en la capa de enrutamiento/control para verificar la correctitud de los servicios de la aplicación.

Creación de proyectos (servicio de proyectos): prueba de la función createProject para garantizar que solo se cree un nuevo proyecto si aún no existe para el usuario. Una función de base de datos simulada verifica si existen proyectos y arroja el error correspondiente si el proyecto ya existe (Figura 33).

```
describe('createProject', () => {
    it('should create a project and return the projectId', async () => {
        const mockProjectId = '64f123456abcdef123456789';
        const projectData = {
            userId: '64f987654abcdef123456789',
            name: 'Test Project',
            description: 'A test project',
            createdAt: new Date(),
            updatedAt: new Date(),
        };

        // Mock the DB response to return no existing project and successfully insert
        db.getProjectByNameAndUserId.mockResolvedValue(null);
        db.createProject.mockResolvedValue(mockProjectId);

        const projectId = await projectService.createProject(projectData);

        expect(projectId).toBe(mockProjectId);
        expect(db.getProjectByNameAndUserId).toHaveBeenCalledWith(projectData.name, projectData.userId);
        expect(db.createProject).toHaveBeenCalledWith(projectData);
    });

    it('should throw an error if project name already exists for the user', async () => {
        const projectData = {
            userId: '64f987654abcdef123456789',
            name: 'Test Project',
            description: 'A test project',
            createdAt: new Date(),
            updatedAt: new Date(),
        };

        // Simulate that a project with the same name already exists for the user
        db.getProjectByNameAndUserId.mockResolvedValue(true);

        // Test that the service throws the correct error
        await expect(projectService.createProject(projectData)).rejects.toThrow(AppError);

        // Ensure that the error thrown is an AppError with type 'CONFLICT'
        try {
            await projectService.createProject(projectData);
        } catch (error) {
            expect(error).toBeInstanceOf(AppError);
            expect(error.type).toBe(AppError.ErrorTypes.CONFLICT);
        }

        // Ensure that db.createProject was not called
        expect(db.createProject).not.toHaveBeenCalled();
    });
});
```

Figura 33. Prueba unitaria de creación de proyectos (servicio de proyectos)

Creación de proyectos (ruta de proyectos): prueba de ruta POST /projects para garantizar que las peticiones sean atendidas correctamente y se responda con mensajes de error esperados para casos esperados. En este caso espera un mensaje de respuesta con código 201 si fue exitoso crear un proyecto nuevo o 403 (conflicto) si se presenta algún conflicto al intentar crear el proyecto, como que un proyecto con ese nombre ya existe (Figura 34).

```
describe('POST /projects', () => {
    it('should create a new project and return the projectId', async () => {
        const mockProjectId = '64f123456abcdef123456789';
        const projectData = {
            name: 'New Project',
            description: 'Test project description',
        };

        // Mock the service response
        projectService.createProject.mockResolvedValue(mockProjectId);

        const response = await request(app)
            .post('/projects')
            .set('Authorization', `Bearer <valid-jwt>`)
            .send(projectData);

        expect(response.statusCode).toBe(201);
        expect(response.body).toEqual({ projectId: mockProjectId });
        expect(projectService.createProject).toHaveBeenCalledWith(expect.objectContaining(projectData));
    });

    it(`should return a ${(new AppError("", AppError.ErrorTypes.CONFLICT)).getHttpStatusCode()} error if project
    creation fails`, async () => {
        const projectData = {
            name: 'New Project',
            description: 'Test project description',
        };

        // Mock the service to throw an error
        projectService.createProject.mockRejectedValue(new AppError(`Project with name ${projectData.name} already
        exists for user with id 1234`, AppError.ErrorTypes.CONFLICT)
        );

        const response = await request(app)
            .post('/projects')
            .set('Authorization', `Bearer <valid-jwt>`)
            .send(projectData);

        expect(response.statusCode).toBe((new AppError("", AppError.ErrorTypes.CONFLICT)).getHttpStatusCode());
    });
});
```

Figura 34. Prueba unitaria de creación de proyectos (ruta de proyectos)

10.1.3 Beneficios de las pruebas unitarias

- **Pruebas aisladas:** las pruebas unitarias permiten centrarse en componentes individuales sin dependencias externas.
- **Retroalimentación más rápida:** dado que se ejecutan en la memoria sin requerir bases de datos ni recursos externos, las pruebas unitarias son rápidas y brindan retroalimentación rápida durante el desarrollo.
- **Calidad de código mejorada:** escribir pruebas unitarias permite escribir código modular y débilmente acoplado, que es más fácil de mantener y refactorizar.

10.2 Pruebas de integración con Postman

10.2.1 Descripción general de las pruebas de integración

Las pruebas de integración verifican que los diferentes componentes del sistema funcionen juntos como se espera. En este proyecto, se utilizó Postman, una herramienta de prueba de API popular, para realizar pruebas de integración en las API de backend. Estas pruebas garantizaron que las rutas, los servicios y las interacciones de la base de datos funcionaran juntos correctamente.

10.2.2 Estrategia de prueba de integración

Postman permitió probar múltiples aspectos de la API:

- Autenticación de usuario: se garantizó que la autenticación basada en JWT funcionara correctamente para el inicio de sesión, el registro y la validación del token.
- Gestión de proyectos y diseño: se probaron puntos finales de API para crear, actualizar, recuperar y eliminar proyectos y diseños, verificando que devolvieran los datos y códigos de estado correctos.
- Manejo de errores: se aseguró de que se devolvieran los mensajes de error y códigos de estado apropiados cuando algo salía mal (por ejemplo, acceso no autorizado, entrada no válida).

10.2.3 Ejemplos de pruebas de integración con Postman

Se hace login con un usuario del sistema (Figura 35)

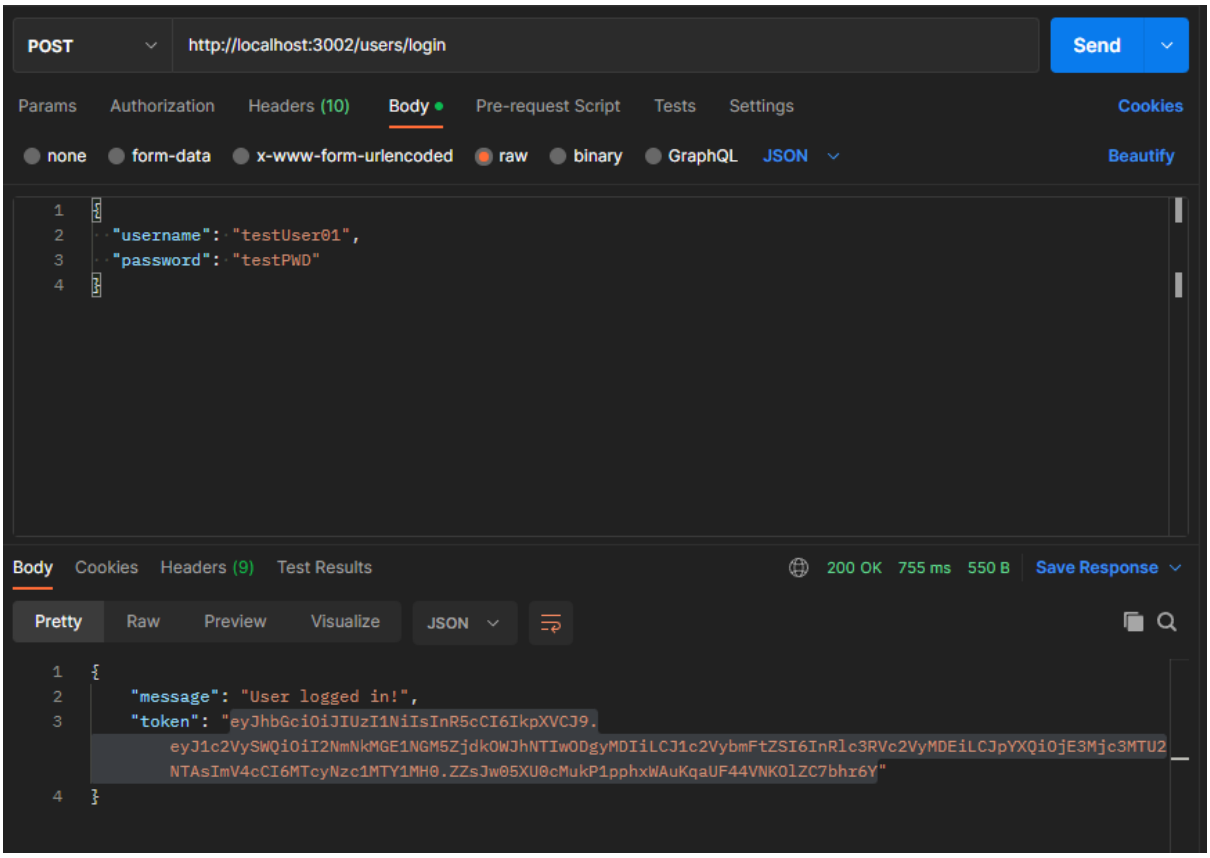


Figura 35. Pruebas de integración con Postman

Se utiliza el JWT para verificar la sesión en los siguientes llamados (Figura 36).

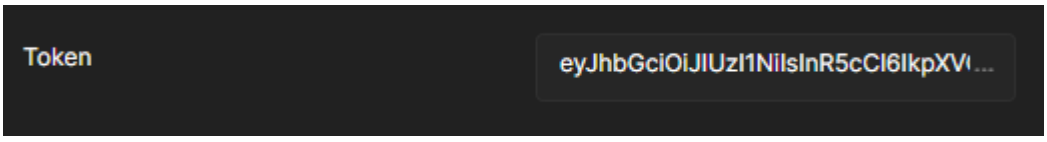


Figura 36. JWT para verificar la sesión en los siguientes llamados

Se prueban los endpoints del proyecto utilizando la autenticación obtenida (Figura 37).

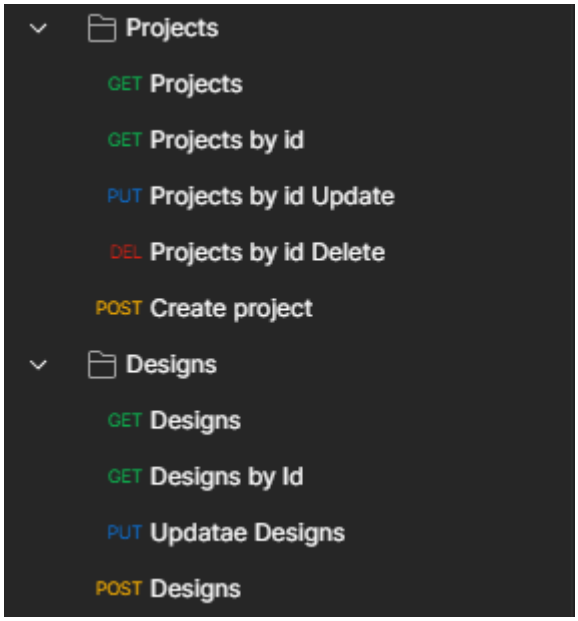


Figura 37. Prueba de autenticación

Se prueban endpoints específicos y escenarios diferentes. En este ejemplo se muestra cómo se prueba el obtener un diseño de un proyecto con los siguientes escenarios:

Escenario 1: En este escenario se muestra una petición hecha con una autenticación válida, de un proyecto válido, para obtener información de un diseño y el resultado es la información y código esperados (Figura 38).

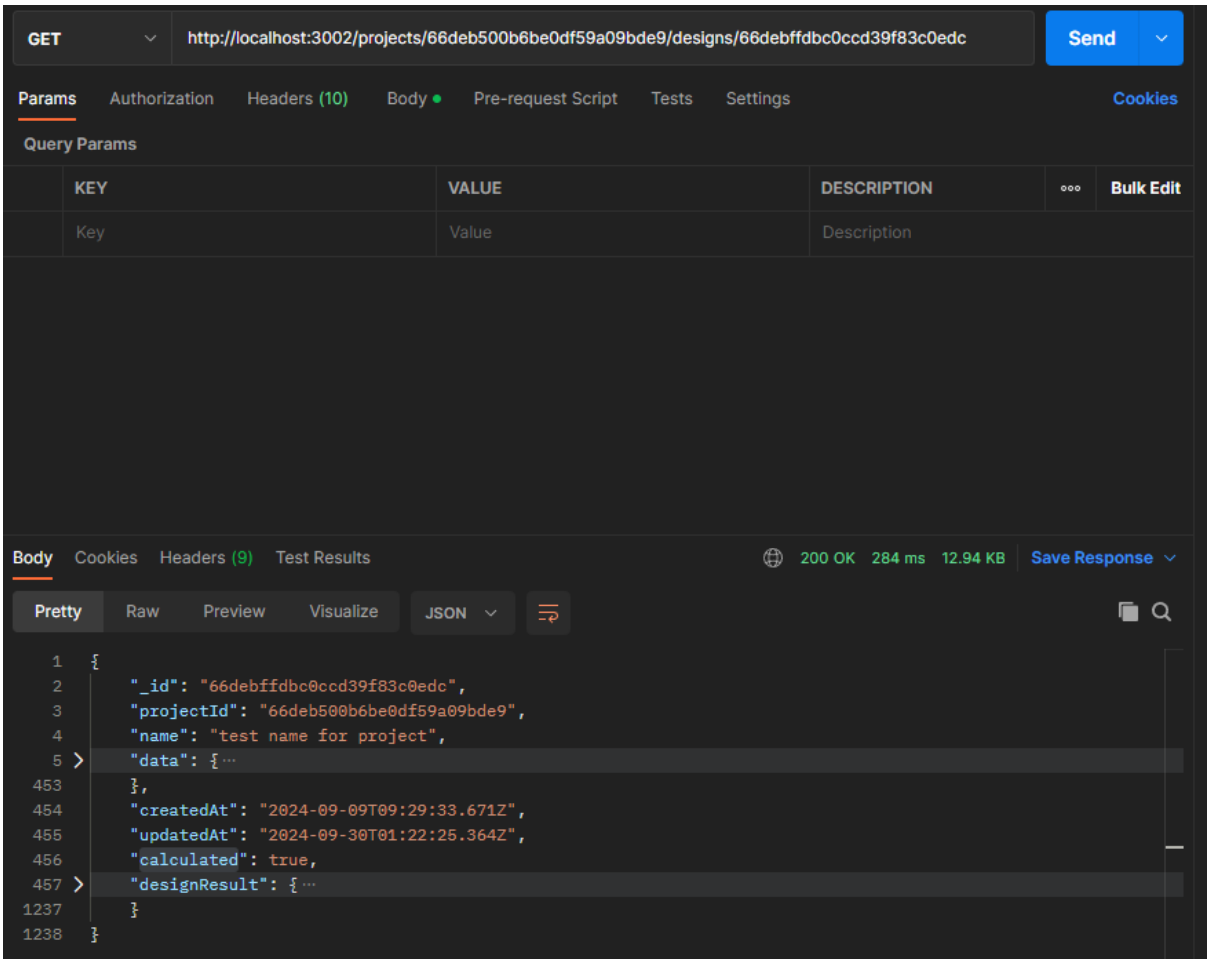


Figura 38. Prueba de escenario 1

Escenario 2: En este escenario se muestra un conflicto con un token de autenticación Invalido (Figura 39).

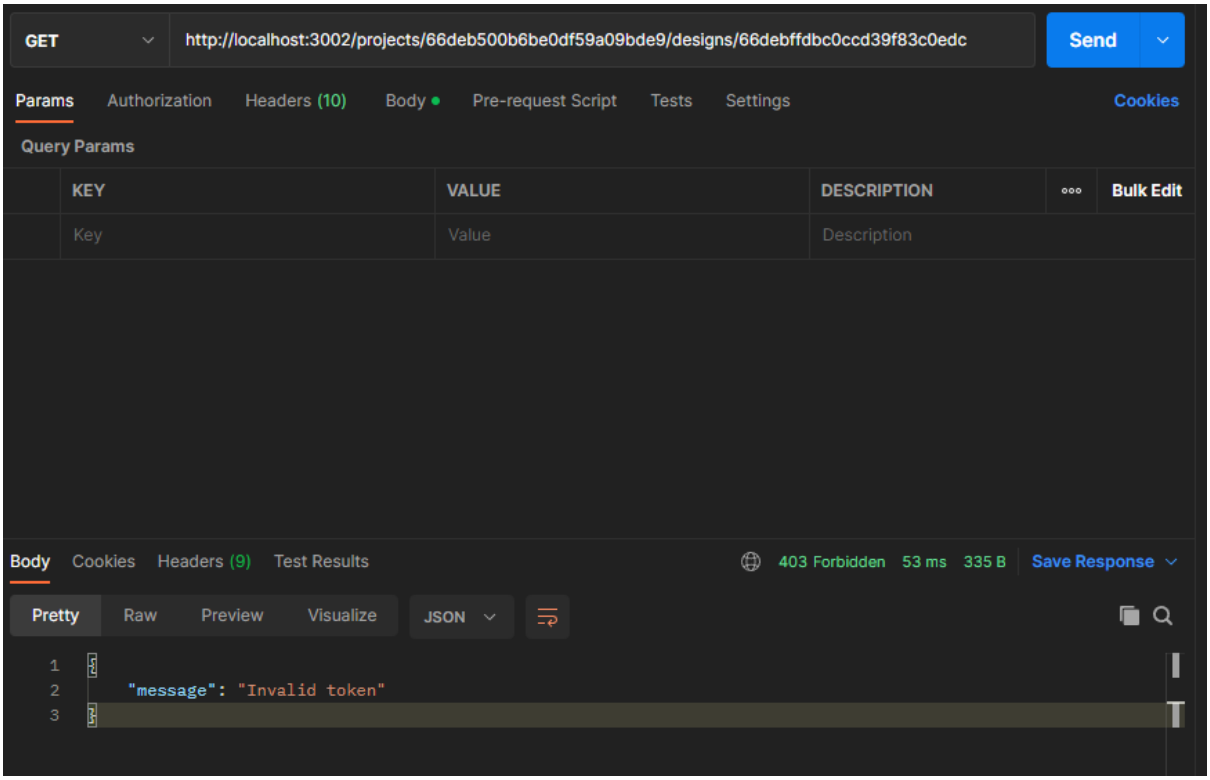


Figura 39. Prueba de escenario 2

Escenario 3: En este escenario se muestra una petición con una autenticación válidada, en un proyecto que pertenece al usuario con un Id de diseño que no existe (Figura 40).

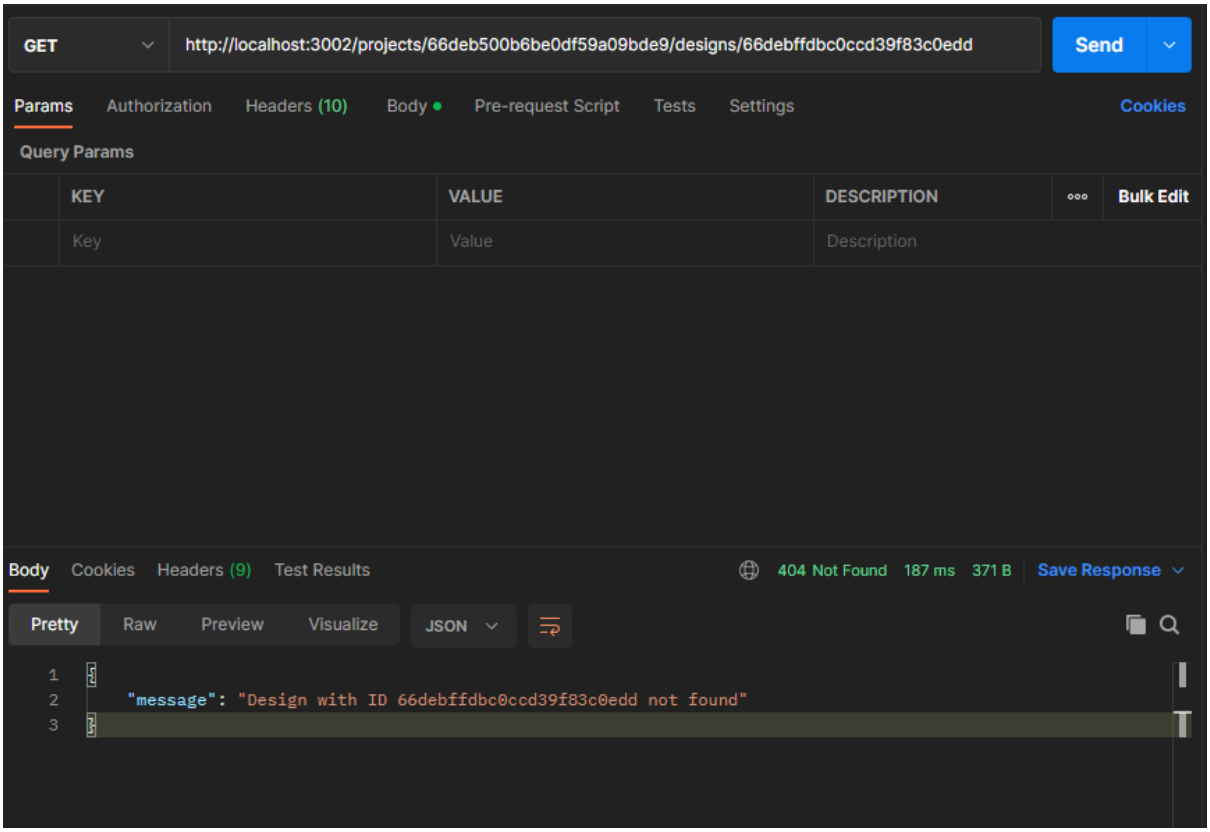


Figura 40. Prueba de escenario 3

Escenario 4: En este escenario se muestra cómo responde ante un error no esperado. En este caso en particular se da un Id de diseño escrito incorrectamente lo cual genera un tipo de error no manejado, pero el servicio es capaz de responder a esto retornando un código 500 correspondiente a internal server error (Figura 41).

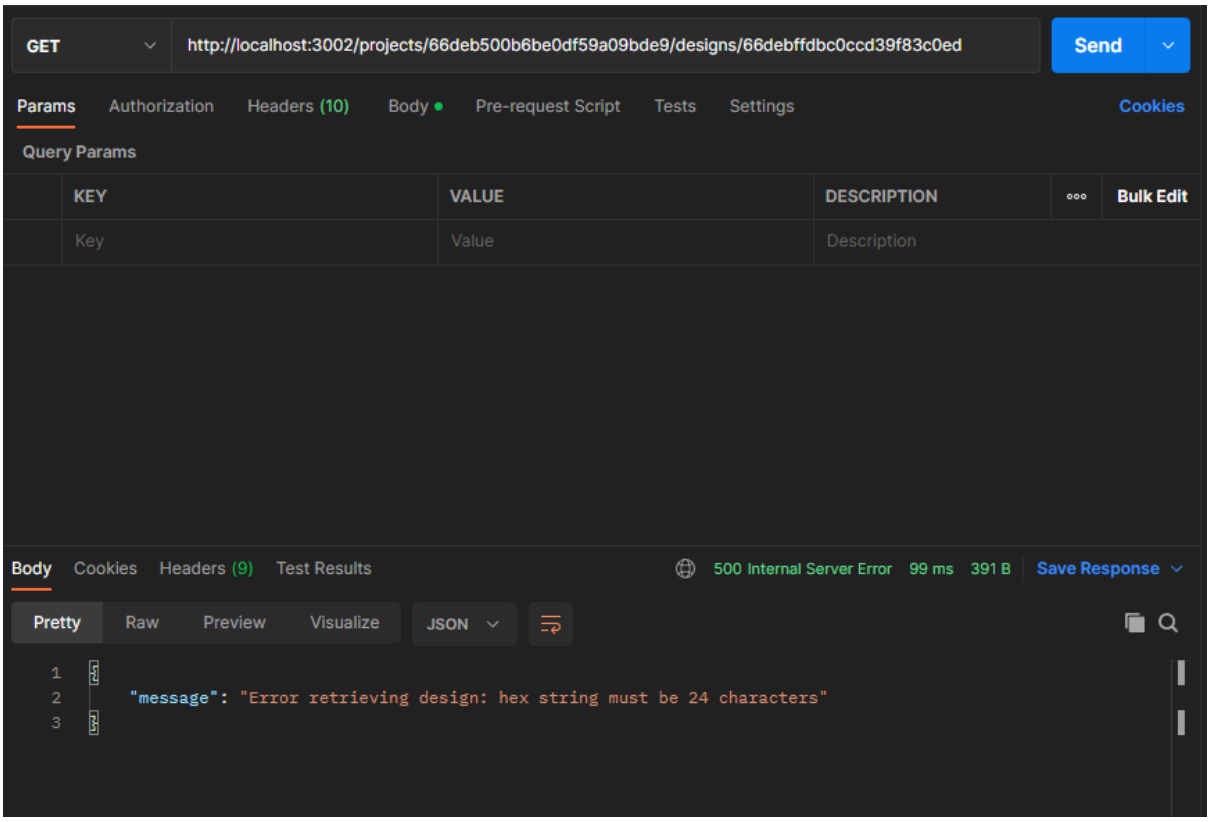


Figura 41. Prueba de escenario 4

Escenario 5: En este escenario se muestra una autenticación válida con Id de proyecto que no existe o no pertenece al usuario (Figura 42).

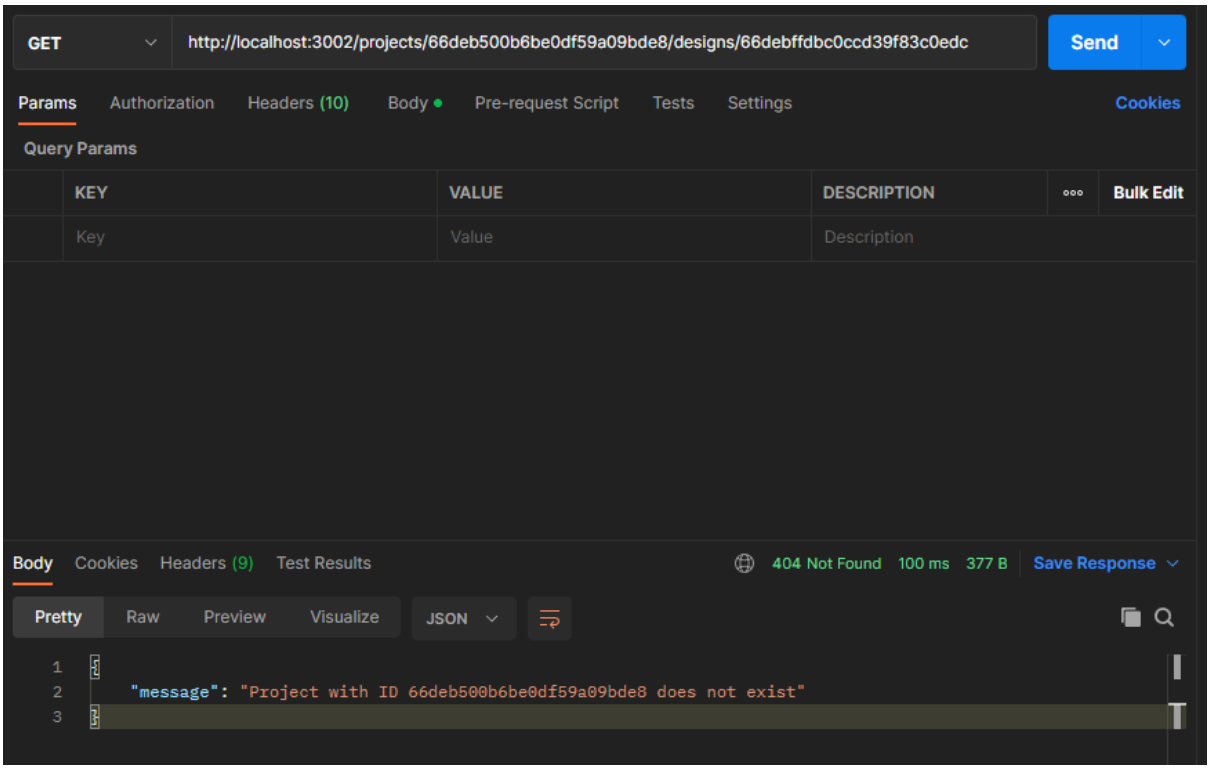


Figura 42. Prueba de escenario 5

10.2.4 Resultados de las pruebas de integración

La fase de pruebas de integración garantizó que las distintas rutas de API interactuaran correctamente con la base de datos y los servicios externos. Las pruebas identificaron problemas como la falta de validación y el manejo incorrecto de errores que se resolvieron durante el desarrollo.

10.3 Pruebas manuales

Las pruebas manuales implican interactuar directamente con la aplicación a través de la interfaz de usuario para verificar que el sistema se comporta como se espera. En este proyecto, las pruebas manuales se realizaron utilizando el frontend de la aplicación donde se simulan las interacciones de los usuarios para asegurar que cada funcionalidad del backlog funcionara correctamente.

10.3.1 Funcionalidades probadas

- **Autenticación de usuario (inicio de sesión/cierre de sesión):** se verificó que los usuarios pudieran iniciar y cerrar sesión en el sistema, con redirección correcta y control de acceso basado en el estado de autenticación.
- **Creación y gestión de proyectos:** se garantizó que los usuarios pudieran crear, editar y eliminar proyectos. También se probó que la interfaz de usuario reflejara actualizaciones en tiempo real después de estas acciones.
- **Gestión de diseño:** se probó el proceso de agregar, actualizar y eliminar diseños dentro de un proyecto. Se verificó que los datos de los formularios se pasaran correctamente al backend y que los resultados se mostraran en el frontend.
- **Visualización de datos:** se aseguró de que los componentes gráficos, como los gráficos que muestran datos del proyecto y del diseño, se actualizaran correctamente según la entrada del usuario. También, se verifico que el grafico que muestra los espectros de carga se actualizara correctamente con el espectro de carga seleccionado proveniente de la base de datos.
- **Manejo de formularios:** se verificó que los datos que ingresara el usuario en los formularios fueran correctos y se le informara al usuario en caso de que no lo fueran.

10.3.2 Resultados de pruebas manuales

Respecto a las pruebas manuales, se realizaron pruebas durante el desarrollo, con cada iteración entregada, donde se probaban las funcionalidades creadas viendo que al usarlas funcionaran como era esperado. También se realizaron pruebas con el cliente mostrándole los cambios y obteniendo retroalimentación que ayudó a que con cada iteración se fuera mejorando la experiencia de usuario general.

Las pruebas manuales ayudaron a:

- Mejorar la manera en la que se usa la aplicación por parte de un usuario final. Se revisa como el usuario interactúa con la aplicación y si esta tiene un flujo cómodo al ser utilizada. Se presta especial atención en donde se ingresan los espectros de carga pues era uno de los requerimientos importantes del sistema.

- Ajustar el diseño en conjunto con los requerimientos y funcionamiento correcto del sistema. Se analizó la disposición y tamaño de la entrada

de datos, los tipos de formularios, el posicionamiento de botones en la interfaz y que datos se deseaban mostrar en los resúmenes.

- Asegurar buen performance de la aplicación dentro de sus casos de uso. Se presta especial atención a como la aplicación responde en escenarios donde se maneja gran cantidad de datos como el formulario de espectro de carga (Figura 28) y el resultado del diseño (Figura 31). Se concluye que tiene tiempos de respuesta correctos.

- Asegurar casos extremos (Edge cases) en los que podría ser utilizada. Se revisan los casos en los que el usuario da los máximos y mínimos en los campos de los formularios y se asegura que cada paso de los cálculos del diseño de resultados correctos.

11. CONCLUSIONES Y RECOMENDACIONES

El primer objetivo específico plateó reconocer la información relevante sobre los desafíos que enfrenta alguien al diseñar un pavimento en Colombia actualmente. A lo largo del proceso de investigación, se identificaron los principales desafíos que enfrentan los ingenieros al diseñar pavimentos en Colombia, los cuales incluyen la falta de acceso a datos actualizados sobre tránsito, las variaciones en los aportes estructurales de las diferentes subbases y las especificidades normativas locales. Estos factores influyen directamente en la durabilidad y sostenibilidad de los pavimentos.

Al reconocer estos desafíos, se pudo estructurar un enfoque de diseño que abordara cada uno de ellos de manera efectiva, integrando variables como el tránsito, los materiales de subbase y los requisitos normativos en la lógica del software desarrollado para optimizar el diseño de pavimentos en distintas regiones del país.

El segundo objetivo específico plateó diseñar un software para esta problemática específica junto con una interfaz sencilla. El diseño del software respondió a las necesidades identificadas en el análisis de la problemática, priorizando la integración de herramientas que simplifiquen la entrada de datos, el análisis del comportamiento del pavimento y la optimización de los recursos en cada proyecto. La interfaz fue concebida para ser intuitiva, minimizando la curva de aprendizaje para los usuarios no especializados en software.

El diseño de una interfaz sencilla permitió que los ingenieros pudieran concentrarse en los aspectos técnicos del diseño de pavimentos sin la distracción de interfaces complejas, garantizando una experiencia de usuario eficiente y accesible.

El tercer objetivo específico plateó Implementar el diseño de software que diera como resultado un producto mínimo viable. La implementación del software culminó en un producto mínimo viable que ofrece todas las funcionalidades esenciales para el diseño de pavimentos, como la entrada de datos sobre tránsito, características de los materiales de subbase y simulaciones de vida útil del pavimento. Esta versión permitió a los usuarios validar el enfoque y evaluar la practicidad del software en proyectos reales.

El producto mínimo viable entregado demostró su capacidad para resolver las problemáticas planteadas del diseño de pavimentos en Colombia, facilitando la toma de decisiones basadas en datos locales del tránsito, un mayor oferta de materiales de subbase y contribuyendo a mejorar la eficiencia del proceso de diseño.

El tercer objetivo específico plateó realizar pruebas de usabilidad y de aceptación. Las pruebas de usabilidad revelaron que la mayoría de los usuarios encontraron el software intuitivo y fácil de usar, confirmando que el enfoque en una interfaz simplificada fue exitoso. Los usuarios lograron completar tareas comunes de diseño con pocos errores y dentro de tiempos razonables.

Las pruebas de aceptación realizadas con ingenieros civiles y expertos en diseño de pavimentos mostraron una alta tasa de satisfacción con el producto. Las recomendaciones surgidas de estas pruebas se integraron en las iteraciones posteriores, mejorando aún más la experiencia del usuario y la precisión del software.

12. BIBLIOGRAFIA

AASHTO. (1993). *AASHTO Guide for Design of Pavement Structures.* Washington, D.C.: U.S. Dept. of Transportation, Federal Highway Administration, National Highway Institute.

AASHTOware. (01 de 07 de 2023). *Pavement*. Obtenido de AASHTOware: https://www.aashtoware.org/products/pavement/pavement-overview/#

ACM. (s.f.). *Association for Computin Machinery*. (ACM) Recuperado el 13 de 3 de 2024, de https://www.acm.org/special-interest-groups/alphabetical-listing

ACPA. (17 de 08 de 2014). *STRETPAVE*. Obtenido de ACPA: https://www.acpa.org/streetpave/

ACPA. (27 de 02 de 2014). *WINPAS*. Obtenido de ACPA: https://www.acpa.org/winpas/

APA. (09 de 06 de 2022). *PavEXpress*. Obtenido de PavEXpress: https://pavexpress.com/session-2-new-pavement-designs-with-1993-and-1998-aashto/

Asociación Colombiana de Ingeniería Sísmica. (2010). *Reglamento Colombiano de Construcción Sismo Resistente NSR-10.* Bogotá: Ministerio de Ambiente Vivienda y Desarrollo Territorial.

Banks, A., Porcello, E., & Porcello, A. (2020). *Learning React: Functional Web Development with React and Redux.* O'Reilly Media, Inc.

Bass, L. P. (2013). *Software Architecture in Practice.* Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., F. M., & ... & Thomas, D. (2001). *Manifesto for Agile Software Development.* Obtenido de Ágile: https://agilemanifesto.org/

Bevacqua, N. (2018). *Practical Modern JavaScript: Dive into ES6 and the Future of JavaScript.* O'Reilly Media, Inc.

Cantelon, M., Harter, T., Holowaychuk, T., & Rajlich, N. (2018). *Node.js in Action.* Manning Publications.

Casciaro, M., & Mammino, L. (2020). *Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques. .* Packt Publishing Ltd.

Casto, J. H., & Orobio, A. (2015). *PCA cálculo*. Obtenido de GRUA: https://sites.google.com/a/correounivalle.edu.co/pcaindustrial/pcac%C3%A1lculo

Chodorow, K. (2013). *MongoDB: The Definitive Guide. .* O'Reilly Media, Inc.

CivilWeb Consulting Engineers. (21 de 01 de 2024). *PCA Pavement Design Spreadsheet*. Obtenido de CivilWeb Spreadsheets: https://civilweb-spreadsheets.com/road-pavement-design/pca-pavement-design-spreadsheet/

Connolly, T. M. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management.* Pearson.

Delatte, N. (2014). *Concrete pavement design, construction, and performance (2nd ed.).* Boca Raton, FL: CRC Press.

Elmasri, R. a. (2015). *Fundamentals of Database Systems.* Pearson.

Federal Highway Administratio. (27 de 06 de 2023). *Pavement*. Obtenido de Federal Highway Administratio: https://www.fhwa.dot.gov/pavement/pavedesign.cfm

FHWA. (May de 2006). *Geotechnical Aspects of Pavements.* Obtenido de U.S. Department of Transportation - Federal Highway Administration : https://www.fhwa.dot.gov/engineering/geotech/pubs/05037/05037.pdf

Gackenheimer, C. (2015). *Introduction to React. .* Apress.

García Aladín, M. F. (2002). *Catálogo de diseño de pavimentos rígidos de la PCA adaptado a las condiciones de tránsito colombianas.* Popayán: Universidad del Cauca.

García Aladín, M. F. (11 de 02 de 2024). La importancia de los computadores en el diseño de pavimentos. (J. F. Otoya García, Entrevistador)

Goel, A., & Das, A. (2021). *Concrete for rigid pavement construction: Recent trends and developments.* Uttar Pradesh, India: Construction and Building Materials, 299.

Griffith, D. (2018). *React Quickly: Painless web apps with React, JSX, Redux, and GraphQL.* Manning Publications.

Griffiths, G., & Thom, N. (2007). *Concrete Pavement Design Guidance Notes.* New York: Taylor & Francis.

Haider, S. W., Harichandran, R. S., & Buch, N. (2021). Characterization of granular and stabilized materials for sustainable pavement design and construction. *Transportation Research Record, 2675(12)*, 108 - 120.

Han, J., & Strickland, J. (2016). *NoSQL Fundamentals.* Packt Publishing Ltd.

Harrison, G. (2015). *Next Generation Databases: NoSQL, NewSQL, and Big Data.* Apress.

Hecht, R., & Jablonski, S. (2011). NoSQL evaluation: A use case oriented survey. In 2011 International Conference on Cloud and Service Computing. *IEEE*, 336-341.

Highsmith, J. (2009). *Agile Project Management: Creating Innovative Products.* Addison-Wesley.

Holmes, D. (2012). *The Enterprise JavaScript Developer. .* Apress.

Huang, Y. H. (2003). *Pavement Analysis and Design, Edition 2.* NJ, USA: Pearson Education.

ICPC, Ministerio de Transporte, INVIAS. (2008). *Manual de diseño de pavimentos de concreto.* Bogotá: Instituto Colombian de Productores de Cemento.

Instituto Mexicano de Transporte (IMT). (19 de 06 de 2022). *IMT-PAVE*. Obtenido de IMT-PAVE: https://imtpave.com/

INVIAS. (2008). *Manual de diseño de pavimentos de concreto para vías con bajos, medios y ltos volúmenes de tránsito.* Bogotá: Ministerio de Transporte, Presidencia de la República.

INVIAS. (2018). *Manual de diseño de pavimentos asfálticos en vías con medios y altos volúmenes de tránsito.* Bogotá: República de Colombia, Ministerio de Transporte.

INVIAS. (10 de 12 de 2023). *Estado de la red vial*. Obtenido de INVIAS: https://www.invias.gov.co/index.php/component/content/article/2-uncategorised/57-estado-de-la-red-vial

Jangda, M., Qadeer, S., Waseem, M., & Karim, A. (2018). Microservices architecture in Node.js. Journal of Computer and Communications, 6(2). 46-58.

Juárez Badillo, E., & Rico Rodríguez, A. (1991). *Mecánica de Suelos_Tomo II.* México: Limusa.

Kappagantula, P., & Kalvakuntla, S. (2016). Full-stack JavaScript development with MEAN. International Journal of Computer Applications, 148(6). 1-5.

Khazanovich, L., & Tompkins, D. (2022). Mechanistic-empirical design of concrete pavements: State of the practice. *Transportation Research Record, 2676(5)*, 71 - 91.

Kniazev, D. (2018). *Vue.js 2 Cookbook: Build modern, interactive web applications with Vue.js.* Packt Publishing Ltd.

Leal Acosta, A. C. (16 de 04 de 2019). *Portafolio*. Obtenido de Colombia requiere 45.000 km de vías adicionales, dice el BID: https://www.portafolio.co/economia/infraestructura/colombia-requiere-45-000-km-de-vias-adicionales-dice-el-bid-528658

Lee, Y.-H., & Carpenter, S. H. (2001). PCAWIN Program for Jointed Concrete Pavement Design. *Tamkang Journal of Science and Engineering*, 293-300.

Maina, J. W., & Luo, X. &. (2022). Advances in characterization and modeling of stabilized granular bases for pavement design. *International Journal of Pavement Engineering, 23(9)*, 2163-2177.

Makin, A. (2017). *Mastering React Test-Driven Development: Build rock-solid, well-tested web apps with React, Redux and GraphQL. .* Packt Publishing Ltd.

Mallick, R. B., & El-Korchi, T. (2018). *Pavement engineering: Principles and practice (3rd ed.).* Boca Raton, FL: CRC Press.

Manifesto for Agile Software Development. (2001). (Agile Alliance) Obtenido de https://agilemanifesto.org/

Manifesto, A. (2001). *Manifesto for agile software development.* Obtenido de https://www.csl.mtu.edu/cs3141/www/notes/Scrum%20[Compatibility%20Mode].pdf

Meier, R. W. (21 de 02 de 2024). *PCA Design Method.* Obtenido de CIVL 4155/6155: (CivilWeb Consulting Engineers, 2024).

Membrey, P., Plugge, E., & Hawkins, T. (2010). *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing.* Apress.

MnDOT. (13 de 05 de 2019). *mnpave*. Obtenido de MnDOT: https://www.dot.state.mn.us/app/mnpave/

Nath, P., Maheshwari, A., & Kumar, M. (2022). Design and performance of rigid pavements: Recent developments and future perspectives. *Journal of Materials Research and Technology, 17*, 2167 - 2183.

Osei, P. (2016). *Learning Node.js Development: Learn the fundamentals of Node.js, and deploy and test Node.js applications on the web.* Packt Publishing Ltd.

Packard, R. G., & Tayabji, S. (1985). New PCA Design Procedure for Concrete Highways and Street Pavements - 1984. *Conference: 3rd International Conference on Concrete Pavement Design* (págs. 225-236). Lafayette, Indiana, USA: Purdue University. Obtenido de Researsh Gate: https://www.researchgate.net/publication/312040049_New_PCA_Design_Procedure_for_Concrete_Highways_and_Street_Pavements_-_1984

Pautasso, C. O. (s.f.). *RESTful Web Services vs. 'Big' Web Services: Making the Right Architectural Decision.* IEEE Internet Computing, vol. 14, no. 1, 2010, pp. 81-85.

Pavementdesigner.org. (2018). *pd*. Obtenido de Pavementdesigner.org: https://www.pavementdesigner.org/projectTypes/street/concrete/projectLevelForm

Porebski, J., Godfrey, K., & Wijngaarde, F. (2011). *JavaScript Web Applications: jQuery Developers' Guide to Moving State to the Client.* O'Reilly Media, Inc.

Portland Cement Association (PCA). (1995). *Thickness Design for Concrete Highway and Street Pavements.* Massachusetts: Portland Cement Association (PCA).

Pressman, R. S. (2015). *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education.

Prozzi, J., & & Hong, F. (2006). *Highway traffic characterization for pavement structure design.* Washington, D.C.: Proceedings of the 85th Annual Meeting of the Transportation Research Board.

Rao, S., Rao, S., Barai, S. V., & Momin, S. S. (2022). Design of rigid pavement for rural roads considering life cycle cost analysis. *Materials Today: Proceedings, 56*, 2087-2091.

Richardson, L. e. (2018). *RESTful Web APIs: Services for a Changing World.* O'Reilly Media.

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. .* Crown Business.

Rozanski, N. a. (2011). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley.

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process. .* Addison-Wesley.

Sadeghi, J. M., Khazanovich, L., & Rodriguez, N. (2022). Mechanistic-empirical structural design of concrete pavement for urban roads. . *Transportation Research Record, 2676(5),*, 92 - 103.

Saha, P., & & Hanaideh, A. (2021). Characterization of stabilized granular base materials for mechanistic-empirical pavement design. . *Transportation Research Record, 2675(12)*, 121-134.

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide.* Obtenido de https://scrumguides.org/scrum-guide.html

Schwartz, A. (2016). *Scaling MongoDB.* O'Reilly Media.

Seo, Y., & Jung, S. (2020). Server-side rendering with React and Node.js. *International Journal of Computer Science and Network Security, 20(4)*, 1-10.

Shaheen, A. A., Tayabji, S. D., & Mounanga, P. (2022). Mechanistic-empirical design procedure for thin concrete pavement. *Transportation Research Record, 2676(4),*, 61 - 70.

Shneiderman, B. e. (s.f.). *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Pearson.

Solano, E. D., & Benavidez, C. A. (28 de 09 de 2022). *Descargas*. Obtenido de Topo3: https://www.topo3.com/descargas/

Sommerville, I. (2016). *Software Engineering.* Pearson.

Tarefder, R. A., & Ahsan, S. (2021). *Design of rigid pavements for sustainable and resilient transportation infrastructure.* Boca Raton, FL: CRC Press.

Tayabji, S., Buch, N., & Wojakowski, J. (2020). *Concrete pavement design guidance and overview. FHWA-HIF-20-002.* WASHINGTON, DC: Federal Highway Administration. .

The designer's corner. (2023). Obtenido de YouTube: https://www.youtube.com/watch?v=PsJiNc_HWE8

Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing, 14(6),* , 80-83.

Timm, D. H. (09 de 2017). *PERROAD*. Obtenido de CESDb.com: https://www.cesdb.com/perroad.html

Wasson, M. (2017). *Angular Development with TypeScript: A complete guide to TypeScript, Angular, and modern Angular development.* O'Reilly Media, Inc.

Wexler, J. (2017). *React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns.* Packt Publishing Ltd.