Manual Técnico

UrbanNav

Juan Esteban Guzmán Henao

Andrés Felipe Ríos Castaño

Jerónimo García Parra

Leidy Johana Nieto Holguín

Profesor: Jefferson Arango López

Ingeniería en Sistemas y Computación

Universidad de Caldas

2023 – Manizales / Villamaría

Tabla de contenido

Introducción

Este manual técnico tiene como objetivo principal describir, explicar y mostrar la aplicación web UrbanNav de manera detallada y comprensible. A través de este documento, se busca proporcionar una visión completa de las bases de la aplicación, para comprender su funcionamiento, las herramientas usadas y la estructura que se utilizará. Este manual no solo servirá como una guía informativa, sino que también permitirá a cualquier persona interesada en su desarrollo la oportunidad de llevarlo a cabo, ya que proporciona los aspectos básicos y necesarios para su implementación.

UrbanNav

Esta plataforma web es una propuesta a las personas como alternativa a viejas conocidas como inDriver, Taxia, Uber, entre otras. Su funcionamiento se basa en la posibilidad de solicitar y gestionar viajes de una manera sencilla y eficiente. Esta aplicación permitirá una interacción inteligente entre los clientes y los conductores, el cual genera un servicio de transporte personalizado a gusto del cliente, ya que está hecha para que la persona que la usa decida tanto las rutas que desea tomar como los conductores en los que quiere ser transportado.

La aplicación tiene las funcionalidades indicadas para que las personas puedan entender fácilmente los usos y pueda solicitar su servicio de transporte de forma placentera. Entre las funciones podemos encontrar:

1. Manejo de algoritmo de Dijkstra, para la de generación de rutas y puntos de inicio y fin.
2. Posibilidad de elegir entre conductores según disponibilidad y según las reseñas que estos lo avalan.

3. Registrarse como cliente o como conductor con sus respectivas funcionalidades según su rol en la aplicación.
4. Capacidad de solicitar un PQRS para sugerencias, preguntas, dudas, entre otras posibilidades.
5. Gestionar su perfil, pudiendo modificar datos personales, cambios en su foto de perfil, preferencias o gustos, validar posibilidad de registrar con Google, autenticación, etc.
6. Calificar tanto clientes como conductores, proporcionándole reseñas, comentarios y calificación numérica.
7. Los clientes podrán acceder a las opiniones de los conductores que les hayan proporcionado sobre sus experiencias de viaje. También podrán consultar su historial de viajes pasados.
8. Se podrá guardar métodos de pago en la aplicación de forma segura para facilitar compras y ahorro de tiempo.
9. Se notificará en todo momento, facturaciones, estado del viaje, confirmaciones de pagos a los clientes vía SMS o correo electrónico
10. Existirá un botón de pánico en el cual se notificará a una persona de su confianza vía SMS o correo electrónico su posición en caso de presentar inconvenientes en el viaje.
11. El administrador dispondrá de un control absoluto sobre los perfiles de los tipos de usuarios, lo que incluye la capacidad de bloquear o activar perfiles de clientes o conductores, siempre proporcionando justificaciones previas. Así mismo, el administrador podrá encontrar rutas, establecer puntos de encuentro entre barrios y ciudades, cálculo de tarifas. Además, tendrá acceso a un conjunto de cinco informes adicionales que le proporcionan información sobre el funcionamiento del sistema y cómo se comporta.

Herramientas de Desarrollo

Utilizaremos herramientas conocidas para lograr un desarrollo efectivo y cómodo para que la plataforma web funcione correctamente, esto mediante uso de frameworks, bases de datos y aplicaciones que nos permitan modificar, probar la aplicación para su testeo, las herramientas en cuestión son:

Node.js:

Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Este será el principal lenguaje de programación que se usará en el proyecto.

Loopback:

Es un "framework" de IBM que integra distintos componentes de Node para poder construir robustas API REST de una manera sencilla y rápida. Con LoopBack podrás interactuar con distintas fuentes de datos a través de modelos, los cuales estarán disponibles para su uso a través de métodos REST.

Loopback permite el manejo y la optimización del backend en nuestra aplicación, así como modificar sus funciones, realizar los microservicios y la lógica de negocio de la plataforma.

Angular:

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

Angular facilita la creación de la interfaz web de la aplicación al permitir la personalización de todos sus elementos. Además, ofrece la flexibilidad necesaria para determinar la disposición de estos elementos y sus respectivas acciones de redirección.

MongoDB:

Es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado.

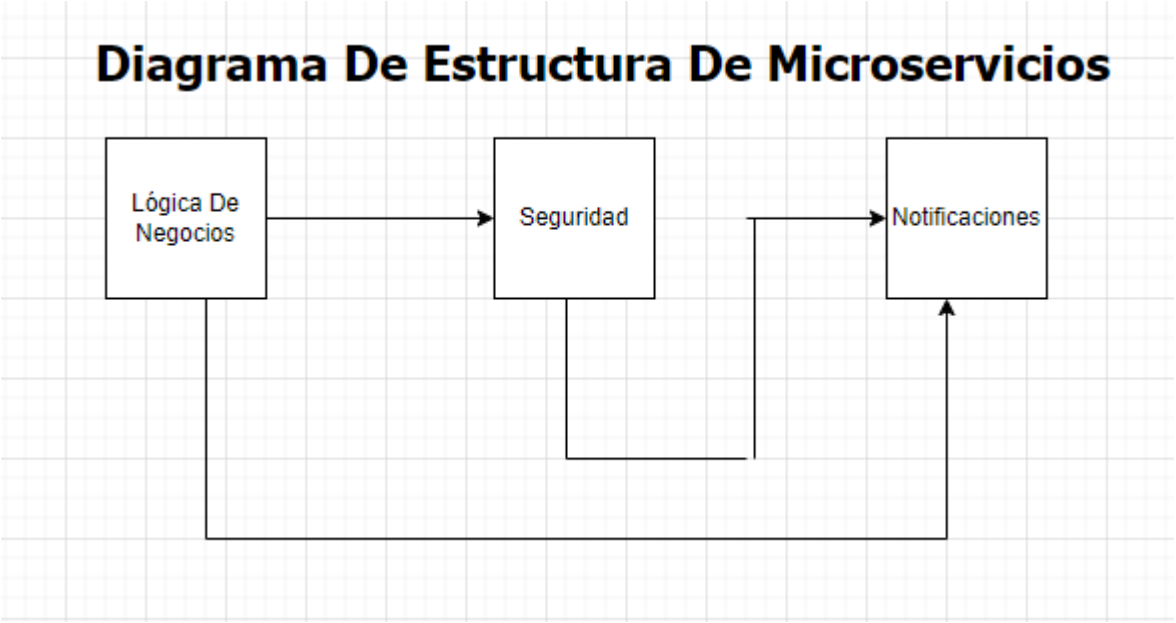
MongoDB será la base de datos que usaremos para guardar los datos que se obtenga de la aplicación, como las reseñas, comentarios, registros de usuario, métodos de pago, roles, entre otras muchas variables.

Mongoose:

Mongoose es una librería para Node.js que nos permite escribir consultas para una base de datos de MongoDB, con características como validaciones, construcción de queries, middlewares, conversión de tipos y algunas otras, que enriquecen la funcionalidad de la base de datos.

La parte central del uso de Mongoose está en la definición de un esquema donde se indica la configuración de los documentos para una colección de MongoDB. Y aunque MongoDB es una base de datos NoSQL, donde los documentos se almacenan sin un esquema predefinido, el uso de un esquema te permite normalizar tu información, sin sacrificar la flexibilidad. Además, hace que la transición de SQL a NoSQL, sea más sencilla.

Diagrama de Arquitectura



La elección de una arquitectura orientada a microservicios sobre una monolítica fue un proceso estratégico basado en una serie de consideraciones cruciales para nuestro proyecto. Esta decisión se fundamenta en varios factores que ofrecen ventajas significativas:

En primer lugar, la escalabilidad es un componente esencial para nuestro proyecto. Los microservicios permiten escalar servicios individuales según las necesidades de carga, lo que nos brinda la flexibilidad de ajustar recursos de manera eficiente y brindar una experiencia óptima a los usuarios.

Además, la simplicidad en el mantenimiento y las actualizaciones fue un factor determinante. En una arquitectura monolítica, cualquier cambio puede afectar a toda la aplicación. Con los microservicios, cada componente es independiente, lo que facilita las actualizaciones sin interrupciones en otros servicios, mejorando la velocidad de implementación y la capacidad de respuesta a las necesidades.

Nuestra elección también se basó en la agilidad del desarrollo. Podemos trabajar en paralelo en diferentes microservicios, acelerando la entrega de características y mejoras.

La flexibilidad en la elección de tecnologías es otro beneficio clave. Cada microservicio puede ser desarrollado utilizando el lenguaje y las herramientas más apropiadas para su tarea específica, lo que nos permite aprovechar las soluciones tecnológicas más adecuadas para cada desafío.

Diagrama de Entidad – Relación

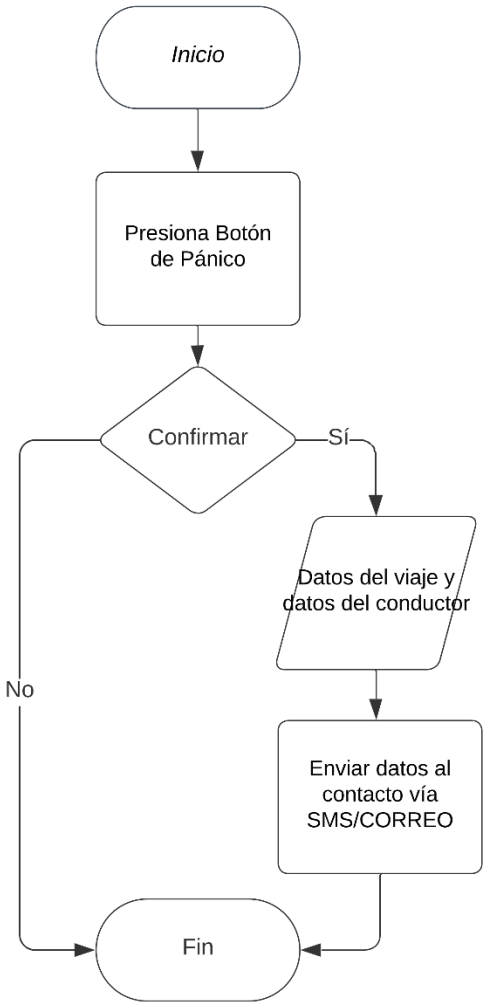
A

Diagrama BPMN

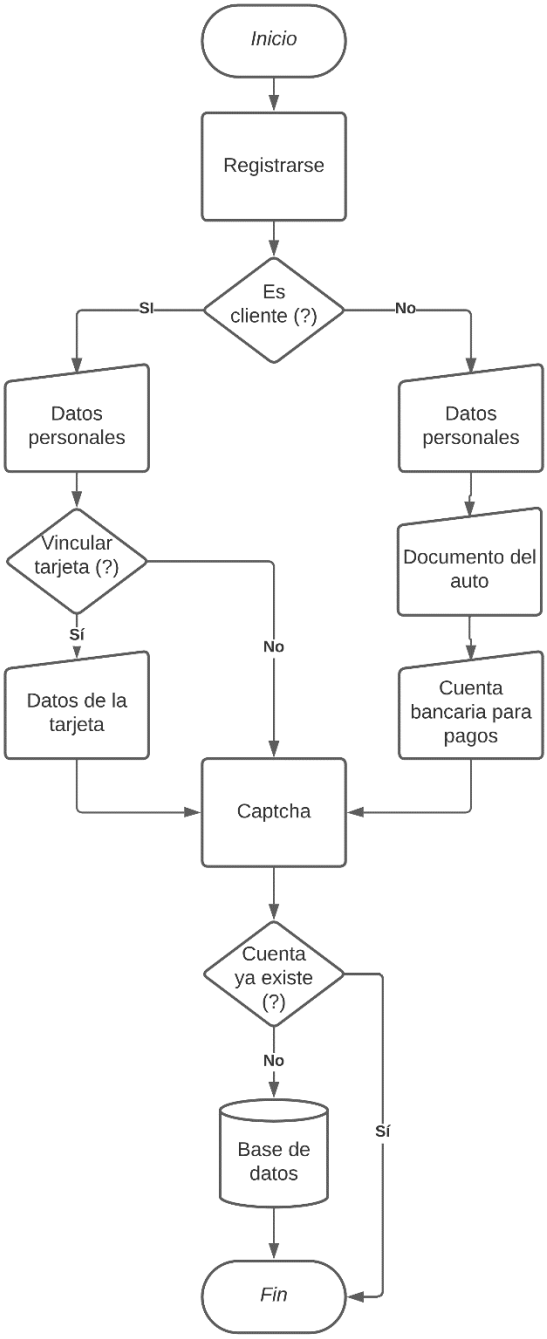
A

Diagramas de Flujo

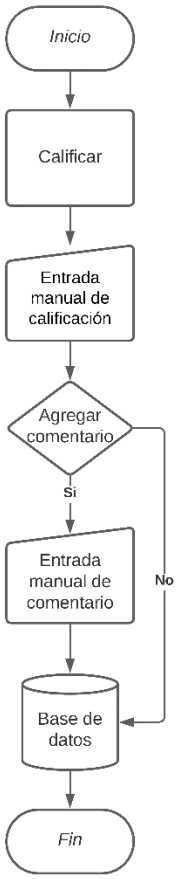
Los diagramas de flujo son herramientas esenciales que nos permiten examinar con mayor detalle las acciones de cada proceso dentro de las funciones de la aplicación. Estos diagramas proporcionan una representación paso a paso de las actividades que se llevan a cabo a medida que avanza el proceso, lo cual reviste una gran importancia. Facilitan enormemente el proceso de codificación al brindarnos una guía clara sobre la implementación, indicando cómo deben desarrollarse las acciones y cuándo deben ejecutarse en momentos específicos del flujo de trabajo.



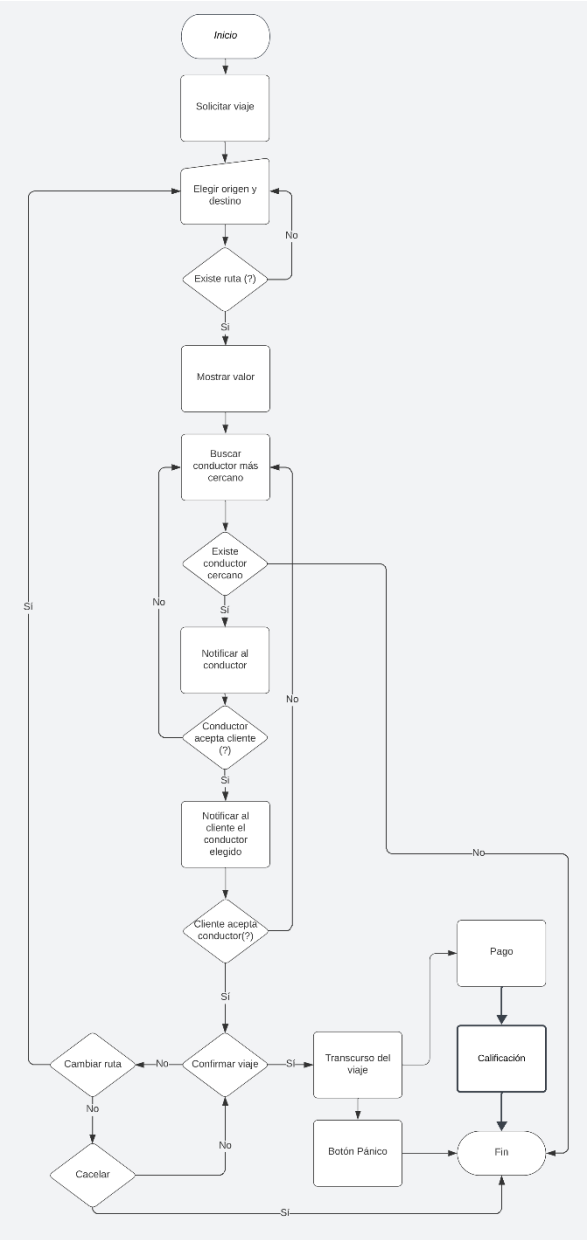
Este diagrama de flujo representa los procesos que se llevaran a cabo al momento de presionar el botón de pánico, se debe confirmar la acción y en caso de que se confirme se procederá a enviar los datos del viaje y del conductor al contacto que el cliente haya seleccionado con anterioridad como “Contacto de Emergencia”, a dicho contacto se le enviara la información vía SMS o por correo electrónico.



Este diagrama de flujo representa la opción de registro en la plataforma, lo que va a preguntar de primero a la persona, es si será un cliente o un conductor, esto es debido a que según el rol que elija se le pedirán datos diferentes, debido que si es cliente, se le pedirán los datos necesarios y un método de pago , pero si es conductor se le exigirá adicionalmente los datos del vehículo que utilizara a la hora de ejercer como conductor y un método donde se le pueda depositar el dinero que genere con los viajes que haga, ya después de diligenciar los datos pertinentes, se hará una prueba captcha que se basa en comprobar que es una persona real la que está creando la cuenta. Finalmente, si pasa la prueba el sistema ya le guardara la cuenta en la base de datos o en caso contrario que el sistema detecte que ya hay una cuenta creada con esos datos, se lo hará saber al que esta haciendo el registro, y ya después de eso finalizaría el proceso.



El diagrama que logramos observar representa como se califica ya sea a un conductor o un cliente, esto funciona de la siguiente manera, primero, después de realizar el viaje al conductor y al cliente les va a llegar un mensaje al teléfono, donde tendrán la opción de calificar su experiencia ya sea con el conductor o el cliente, entonces si se decide aceptar, lo que sucederá es que la persona podrá ponerle de 1 a 5 estrellas y dejar un comentario con respecto a la elección de esas estrellas, después se envía el comentario y finalmente se guarda el comentario en la base de datos de la persona, ya sea del cliente o conductor, y así finalizaría el proceso.



Vamos ahora con el diagrama de flujo mas complejo de la lista debido a que este tiene bastantes opciones a tener en cuenta, este diagrama representa la solicitud de un viaje, para comenzar, se solicita la opción de emprender un viaje, lo primero que hace el sistema es identificar su posición y una posible posición cerca para establecerla como punto de inicio, después, la persona elije el destino final del trayecto, y el sistema empezara a calcular una ruta en la que se pueda emprender un viaje, si esto no se cumple, no se podrá realizar el viaje, en caso dado de que si, ahora el sistema buscara el conductor disponible mas cercano, si lo encuentra le llegara una notificación al conductor si desea llevar a este cliente, si el conductor no acepta, el sistema buscara otro conductor al cliente, si lo hace, inmediatamente le llegara una notificación al cliente diciendo que este conductor lo piensa llevar a su destino, ahí entra si el cliente decide aceptar a ese conductor, si no lo hace empieza a buscar otro conductor, en caso dado de que si, entonces aparece otro anuncio si confirmar finalmente el viaje, si el cliente no lo hace, se le preguntara si quiere cambiar la ruta, y si dice que si, se devuelve al inicio, si no cambia de ruta, entonces se finaliza el proceso, en caso de que se confirme el viaje, se procede con el viaje y ya queda la opción si se recurre al botón de pánico, si todo sale bien, se procede con el pago y la calificación de ambas partes y finaliza el proceso.