Índice

Diagrama de arquitectura de la solución e Infraestructura.

Se plantea una solución basada en la nube de AWS, consultar el Anexo 1 para ver el diagrama que se detalla a continuación.

1. Se usa CloudFront para tener alta disponibilización del front, CloudFront agrega encabezados de seguridad de transporte estricta HTTP (HSTS), que agrega seguridad para cada respuesta de la distribución. Un bucket de Amazon Simple Storage Service (Amazon S3) aloja la interfaz de usuario web (IU web).
2. Amazon CloudFront dispara una función Lambda desplegada en AWS Edge, Lambda@Edge actúa como autorizador para validar las peticiones del cliente, que usa un token de acceso incluido en el

authorization header de HTTP. El token es validado con Amazon Cognito.

3. Un bucket de Amazon Simple Storage Service (Amazon S3) aloja la interfaz de usuario mobile (IU mobile), AWS Amplify toma los recursos de allí para disponibilizar la app mobile.
4. AWS WAF protege la API de AWS AppSync de los exploits y bots comunes que pueden afectar la disponibilidad, poner en riesgo la seguridad o consumir demasiados recursos.
5. Los puntos de conexión de AWS AppSync permiten que el componente de la IU web solicite datos de relación de recursos, consulte costes e importe nuevas regiones de AWS.
6. AppSync usa Resolvers para hacer diferentes llamados directos a diferentes servicios en VPCs.
7. El balanceador de cargas distribuye el tráfico entre varias zonas de disponibilidad. Y Consulta

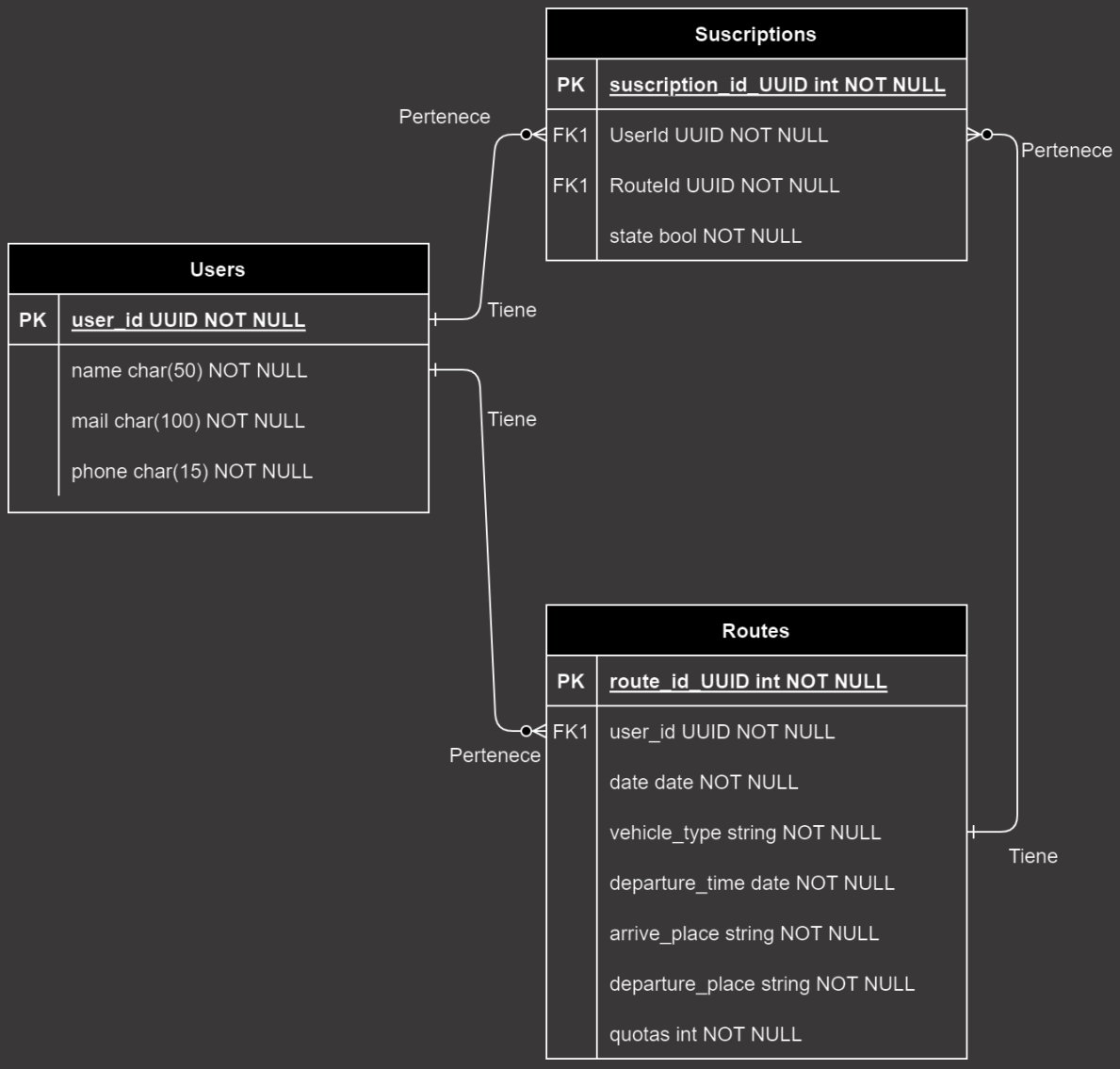
los recursos de los servicios que se encuentran en máquinas EC2.

8. AWS Directory Service le permite ejecutar Microsoft Active Directory (AD) como un servicio administrado. Se crea como un par de controladores de dominio de alta disponibilidad conectados a VPC. Los controladores de dominio se ejecutan en diferentes zonas de disponibilidad. EC2 contiene los servicios que se conectan con AD para la disponibilización y manipulación de datos.
9. Amazon ECR contiene una imagen de Docker con el servicio. Amazon ECS administra la tarea de AWS EC2 y proporciona la configuración necesaria para ejecutar la tarea.
10. La función coste de Lambda usa Amazon Athena para consultar Informes de costos y usos de (AWS

CUR) para proporcionar datos de costos estimados a la aplicación.

11. Athena ejecuta consultas en CUR.
12. CUR entrega los informes al bucket CURBucket de S3.
13. La función coste de Lambda almacena los resultados de Athena en el bucket AthenaResultsBucket de S3.

Modelo Entidad Relación



Tecnologías

React JS y React Native

Desarrollo Multiplataforma: React JS y React Native permiten el desarrollo de aplicaciones web y móviles utilizando un único código base. Dado que se espera una aplicación móvil y una página web, esta capacidad de desarrollo multiplataforma es crucial para reducir el esfuerzo de desarrollo y el tiempo de entrega.

Interfaz de Usuario Unificada: La interfaz de usuario basada en React permite una experiencia de usuario consistente tanto en la aplicación móvil como en la web, lo cual es importante para la usabilidad y la adopción del usuario.

Rendimiento: React Native utiliza una arquitectura nativa que proporciona un rendimiento comparable al desarrollo nativo. Esto es crucial para una aplicación que puede tener una gran cantidad de usuarios concurrentes.

Nest

Escalabilidad: Nest es un framework de Node.js que se enfoca en la escalabilidad y modularidad. Dado que se espera un número considerable de usuarios (500 diarios), Nest proporciona una estructura que facilita la escalabilidad horizontal y vertical.

Integración con Directorio Activo: Nest puede integrarse fácilmente con servicios de autenticación como el

Directorio Activo, lo que facilita la gestión de la autenticación y autorización de los usuarios de la compañía.

API RESTful: Nest permite la creación de APIs RESTful, lo cual es ideal para la comunicación entre la aplicación frontend y el backend. Esto facilita la implementación de las funcionalidades requeridas.

DynamoDB

Escalabilidad y Rendimiento: DynamoDB es un servicio de base de datos NoSQL altamente escalable y de alto rendimiento. Dado el número esperado de usuarios y la necesidad de un rendimiento eficiente, DynamoDB puede manejar fácilmente la carga de trabajo y escalar según sea necesario.

Integración con AWS: Al ser un servicio de Amazon Web Services (AWS), DynamoDB se integra de manera natural con otras herramientas y servicios de la nube de AWS, proporcionando una solución integral y fácil de mantener.

Flexibilidad de Datos: La estructura flexible de DynamoDB permite almacenar y recuperar datos de manera eficiente, lo que es esencial para la variedad de datos relacionados con las rutas y los usuarios.

En resumen, la combinación de React JS y React Native para el frontend, Nest para el backend, y DynamoDB como base de datos proporciona una solución eficiente,

escalable y fácil de mantener para la aplicación de gestión de rutas de transporte, cumpliendo con los requisitos de la compañía en términos de funcionalidad, rendimiento y experiencia del usuario.

Metodología de desarrollo

Se practicaría la metodología Scrum para centrarse en la entrega iterativa e incremental del producto. Aquí se describe cómo se podría aplicar Scrum al proyecto de la aplicación de gestión de rutas de transporte, junto con los roles de equipo asociados.

Roles de Equipo:

Product Owner

Define y prioriza el backlog del producto, identificando las características clave y las historias de usuario.

Colabora con los stakeholders para comprender y refinar los requisitos del producto.

Participa en las reuniones de planificación y revisión de sprint.

Scrum Master

Facilita y apoya las actividades del equipo Scrum, eliminando obstáculos y promoviendo la colaboración.

Asegura que el equipo siga las prácticas y valores de Scrum.

Organiza y facilita las reuniones diarias, de planificación, revisión y retrospectiva.

Equipo de Desarrollo

Compuesto por desarrolladores, arquitectos de software y testers.

Trabajan en conjunto para entregar las historias de usuario seleccionadas durante el sprint.

Se autoorganizan para determinar cómo cumplir con los objetivos del sprint.

Proceso Scrum:

Planificación del Sprint

El Product Owner presenta las historias de usuario prioritarias.

El equipo de desarrollo selecciona historias que pueden completarse durante el sprint.

Se establece el objetivo del sprint y se comprometen con la cantidad de trabajo seleccionada.

Desarrollo del Sprint

El equipo trabaja en las historias de usuario seleccionadas.

Se llevan a cabo reuniones diarias cortas (Daily Standup) para compartir el progreso y discutir obstáculos.

El Scrum Master elimina obstáculos y facilita la colaboración.

Revisión del Sprint

Al final del sprint, el equipo presenta las funcionalidades completadas al Product Owner y a los stakeholders.

Se revisa el trabajo realizado y se recopila feedback para ajustar futuros sprints.

Retrospectiva del Sprint

El equipo reflexiona sobre el sprint, identifica áreas de mejora y define acciones para implementar en el próximo sprint.

El Scrum Master facilita la retroalimentación y ayuda al equipo a mejorar continuamente.

Artefactos Scrum:

Product Backlog

Lista priorizada de todas las funcionalidades, historias de usuario y tareas pendientes para el producto.

Mantenido por el Product Owner y actualizado regularmente.

Sprint Backlog

Lista de las historias de usuario seleccionadas para el sprint y las tareas asociadas.

Mantenido por el equipo de desarrollo y utilizado para hacer un seguimiento del progreso durante el sprint.

Incremento del Producto

La suma de todas las historias de usuario completadas durante el sprint.

Debe ser potencialmente entregable y mejorar la funcionalidad del producto.

Ciclo de Scrum:

Sprint

Un período de tiempo fijo durante el cual se lleva a cabo el trabajo (usualmente 2 a 4 semanas).

Al final de cada sprint, se produce un incremento del producto.

Reuniones Diarias (Daily Standup)

Breves reuniones diarias donde los miembros del equipo comparten actualizaciones sobre el progreso y discuten obstáculos.

Reunión de Planificación del Sprint

Al inicio de cada sprint, se seleccionan las historias de usuario y se planifica el trabajo.

Reunión de Revisión del Sprint

Al final de cada sprint, se presenta el trabajo completado y se recopila feedback.

Reunión de Retrospectiva del Sprint

Al final de cada sprint, el equipo reflexiona sobre su desempeño y planifica mejoras.

Al aplicar Scrum, el equipo puede adaptarse a cambios en los requisitos, mejorar continuamente y entregar un producto de alta calidad de manera incremental.

Buenas Practicas

Metodología del Proyecto:

Historias de Usuario

Definir historias de usuario claras y específicas que reflejen los casos de uso de los empleados, priorizando las características más críticas para la funcionalidad principal.

Iteraciones Cortas

Dividir el desarrollo en iteraciones cortas y entregables. Realiza revisiones regulares con el equipo y los stakeholders para obtener retroalimentación temprana.

Pruebas Continuas

Implementar pruebas unitarias, de integración y end-to-end de manera continua durante el desarrollo para garantizar la estabilidad y la funcionalidad del sistema.

Despliegue Continuo

Automatizar el proceso de despliegue para facilitar la entrega continua y minimizar los errores en producción.

Revisión de Código

Realizar revisiones de código entre los miembros del equipo para garantizar la coherencia, calidad y conformidad con las mejores prácticas.

Gestión de Problemas

Utilizar herramientas de gestión de problemas y seguimiento, como Jira, para rastrear tareas, errores y mejoras.

Código:

Estructura de Proyecto Organizada

Mantener una estructura de proyecto clara y organizada para facilitar la navegación y la comprensión del código.

Modularidad

Dividir el código en módulos y componentes reutilizables para promover la modularidad y facilitar el mantenimiento.

Nombres Significativos

Utiliza nombres de variables, funciones y clases significativos y descriptivos para mejorar la legibilidad del código.

Documentación

Crear una documentación clara para que otros desarrolladores puedan entender rápidamente el propósito y el funcionamiento de cada parte del código.

Gestión de Dependencias

Utilizar un sistema de gestión de dependencias, como npm o yarn, para administrar y versionar las bibliotecas utilizadas en el proyecto. No actualizar todos los paquetes de manera automática para así tener mejor control de versiones de dependencias.

Seguridad

Implementar medidas de seguridad adecuadas, como la validación de entrada, protección contra ataques de inyección y la gestión segura de las credenciales de autenticación.

Optimización de Rendimiento

Optimizar el rendimiento del código, especialmente en el lado del frontend, mediante la carga diferida, el manejo eficiente de recursos y la minimización de solicitudes de red.

Control de Versiones

Utilizar un sistema de control de versiones, como Git, y seguir prácticas como ramificación y fusiones regulares para gestionar eficientemente las actualizaciones y correcciones.

Internacionalización y Localización

Considera la internacionalización y localización desde el principio, permitiendo que la aplicación pueda adaptarse a diferentes idiomas y regiones.

Monitorización y Registro

Implementa registros detallados y herramientas de monitorización para facilitar la identificación y solución rápida de problemas en producción.

Riesgos y mitigación

Retrasos en el Desarrollo

Riesgo: La falta de experiencia con las tecnologías seleccionadas o problemas imprevistos pueden resultar en retrasos en el desarrollo.

Mitigación: Realizar evaluaciones de habilidades del equipo al inicio del proyecto. Proporcionar capacitación cuando sea necesario y asignar recursos con experiencia en las tecnologías clave.

Cambios en los Requisitos

Riesgo: Cambios constantes en los requisitos del proyecto pueden afectar la planificación y la entrega.

Mitigación: Establecer una comunicación clara con el Product Owner y stakeholders. Utilizar metodologías ágiles para adaptarse a cambios y realizar revisiones frecuentes del backlog.

Problemas de Integración

Riesgo: Problemas al integrar los componentes frontend y backend, especialmente al trabajar con tecnologías diferentes.

Mitigación: Realizar pruebas de integración continuas y utilizar herramientas de integración continua. Establecer

una interfaz de contrato claro entre los equipos de frontend y backend.

Problemas de Escalabilidad

Riesgo: La aplicación puede no escalar adecuadamente para manejar el aumento esperado en el número de usuarios.

Mitigación: Realizar pruebas de carga y rendimiento anticipadas para identificar cuellos de botella. Utilizar servicios escalables de AWS como EC2, dynamodb y cloudfront, y ajustar la infraestructura según sea necesario.

Problemas de Seguridad

Riesgo: Vulnerabilidades de seguridad, especialmente cuando se manejan datos sensibles de los empleados.

Mitigación: Implementar prácticas de seguridad desde el diseño, realizar pruebas de seguridad regulares, utilizar servicios de AWS (como amazon cognito, o amazon WAF) con controles de seguridad integrados y seguir las mejores prácticas de seguridad de AWS.

Problemas de Cumplimiento Normativo

Riesgo: No cumplir con las regulaciones de privacidad o cumplimiento normativo que puedan aplicar a la aplicación.

Mitigación: Mantenerse informado sobre las regulaciones locales y sectoriales, implementar medidas de privacidad y seguridad, y realizar auditorías periódicas para garantizar el cumplimiento.

Problemas de Disponibilidad

Riesgo: Interrupciones en los servicios, afectando la disponibilidad de la aplicación.

Mitigación: Utilizar servicios de AWS con acuerdos de nivel de servicio (SLA) garantizados. Implementar redundancia (zonas de disponibilidad) y respaldos regulares.

Desviación del Presupuesto

Riesgo: Gastos imprevistos pueden llevar a una desviación del presupuesto.

Mitigación: Realizar una planificación financiera cuidadosa. Monitorear y ajustar el presupuesto a medida que evoluciona el proyecto. Utilizar servicios de AWS de manera eficiente para optimizar costos (ver componente de costos en el diagrama de arquitectura).

Problemas de Comunicación

Riesgo: Falta de comunicación entre los miembros del equipo, stakeholders y otras partes interesadas.

Mitigación: Establecer canales de comunicación claros, realizar reuniones regulares de seguimiento y fomentar la transparencia en la comunicación.

Problemas de Cumplimiento con el Manual de Marca

Riesgo: No cumplir con las directrices del manual de marca de la compañía puede afectar la aceptación del usuario.

Mitigación: Involucrar a expertos en diseño y seguir cuidadosamente las directrices de la marca en el desarrollo de la interfaz de usuario.