PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

ARQUITETURA E SOLUÇÕES CLOUD

BRUNO MARQUES

JOÃO ELIAS FADEL

JOSÉ MUSSY

JOÃO VICTOR FERREIRA

LOHINE MUSSI

TDE 3

CURITIBA

2024

About arc42

arc42, the template for documentation of software and system architecture.

Template Version 8.2 EN. (based upon AsciiDoc version), January 2023

Created, maintained and © by Dr. Peter Hruschka, Dr. Gernot Starke and contributors. See https://arc42.org.

This version of the template contains some help and explanations. It is used for familiarization with arc42 and the understanding of the concepts. For documentation of your own system you use better the *plain* version.

Link do projeto: https://github.com/take-a-guide

Estudantes: Bruno Marques, João Elias Fadel, José Mussy, João Victor Ferreira e Lohine Mussi

1. Introdução e Metas

O desenvolvimento da plataforma "Take-A-Guide" é impulsionado por uma série de requisitos e forças direcionadoras que são cruciais para os arquitetos de software e a equipe de desenvolvimento. Esses fatores incluem:

- Objetivos de Negócio: Democratizar o acesso ao turismo, proporcionando uma experiência de viagem personalizada e eficiente que atenda às necessidades específicas dos turistas e maximize a exposição dos guias turísticos a um público mais amplo.

- Funcionalidades Essenciais: Oferecer uma plataforma intuitiva que permita a turistas pesquisar, selecionar e contratar guias turísticos especializados em diversos nichos, como história, gastronomia e natureza. Possibilitar a personalização de itinerários e a reserva de tours diretamente pelo sistema.

- Requisitos Funcionais Essenciais: Implementar funcionalidades que permitam a criação e gestão de perfis de guias, visualização de avaliações e classificações de outros turistas, agendamento de tours, e processamento seguro de pagamentos.

- Metas de Qualidade para a Arquitetura: Garantir a escalabilidade para suportar um número crescente de usuários, tanto turistas quanto guias. Assegurar a segurança e a privacidade dos dados dos usuários. Oferecer alta disponibilidade e desempenho, proporcionando uma experiência de usuário fluida e confiável.

-

1.1 Visão Geral dos Requisitos

Conteúdo

Esta seção oferece uma breve descrição dos requisitos funcionais e das forças direcionadoras do desenvolvimento da plataforma "Take-A-Guide". O foco é proporcionar um entendimento claro dos objetivos principais e funcionalidades essenciais da plataforma.

Motivação

O sistema "Take-A-Guide" foi desenvolvido com o objetivo de melhorar a experiência de viagem personalizada dos turistas e aumentar a visibilidade e oportunidades de negócio para os guias turísticos. A plataforma visa facilitar a conexão entre turistas e guias, tornando o processo de agendamento de tours mais eficiente e seguro, além de democratizar o acesso ao turismo de qualidade.

- **Requisitos Funcionais**

ID	Descrição	Prioridade
RF1	O sistema deve permitir o cadastro de usuários para as seguintes personas: contratante, guia e administrador.	Alta
RF2	O sistema deve permitir as personas realizarem o login na plataforma.	Alta
RF3	O sistema deve garantir a segurança dos dados dos usuários e conformidade com as normas de privacidade e proteção de dados.	Alta
RF4	O sistema deve oferecer uma maneira segura de realizar pagamentos online para reservas de tours.	Alta
RF5	O sistema deve permitir diferentes níveis de acesso, variando de acordo com o tipo de usuário, sendo eles guias, turistas e administrador.	Alta
RF6	O sistema deve permitir que o contratante possa agendar e reservar passeio diretamente pela plataforma.	Alta
RF7	O sistema deve permitir que o contratante busque guias turísticos com base em diferentes critérios, como localização, nicho de interesse (história, gastronomia, natureza, etc.), e avaliações de outros usuários.	Alta
RF8	O sistema deve permitir recuperar usuário e/ou senha para todas as personas	Alta
RF9	O sistema deve ter uma funcionalidade para permitir que o usuário possa solicitar a verificação para se tornar um guia	Alta

RF10	O sistema deve permitir que os perfis de guias turísticos exibam informações detalhadas, incluindo biografia, especializações, avaliações, classificações de outros turistas por estrelas (de 0 a 5) e todos os passeios que estão sendo ofertados, além de fotos e vídeos, se disponíveis.	Alta
RF11	O sistema deve permitir que os guias insiram uma postagem sobre um determinado passeio que ele deseja realizar, contendo descrição da atividade, valor, tags e datas e horários disponíveis para reserva.	Alta
RF12	O sistema deve permitir que o Guia possa editar os seus anúncios alterando número de vagas, datas e horários disponíveis.	Alta
RF13	O sistema deve permitir que o administrador avalie o KYC (Know Your Customer) do guia na plataforma.	Média
RF14	O sistema deve permitir que o contratante deixe avaliações em comentários e classificações de experiência após a realização dos tours. A avaliação deverá conter classificação por estrelas (de 0 a 5), e um input para comentário.	Média
RF15	O sistema deve permitir que os usuários possam editar seus perfis, sendo elas informações pessoais, como foto de perfil, biografia, especializações e preferências.	Média
RF16	O sistema deve permitir que no ato da reserva o contratante possa enviar uma mensagem ao guia para indicar quais aspectos têm mais interesse.	Baixa

RF17	O sistema deve permitir que o Contratante possa realizar o cancelamento de uma reserva.	Baixa
RF18	O sistema deve disparar um e-mail de notificação para o guia quando o pagamento de uma reserva é realizado	Baixa
RF19	O sistema deve disparar um e-mail de notificação para o contratante quando o pagamento da reserva é realizado, informando data, horário, guia e local de encontro.	Baixa

RF20	O sistema deve oferecer um chat para facilitar a comunicação entre turistas e guias turísticos antes e após a reserva de um tour.	Baixa
RF21	O sistema deve realizar o update da conta do guia para super guia automaticamente, caso atenda às regras de negócio.	Média

Tabela 1. Requisitos Funcionais.

- **Requisitos Não Funcionais**

RNF1	O sistema deve seguir as leis de proteção de dados (LGPD).	Alta
RNF2	O sistema deve utilizar um gateway para realizar as transações dos pagamentos	Alta
RNF3	O front-end deve ser desenvolvido em Next.js	Alta
RNF4	O sistema deve ser desenvolvido utilizando o Framework Java Spring Boot	Baixa
RNF5	O sistema deverá ser hospedado em cloud.	Alta
RNF6	O sistema deve ter um tempo de resposta rápido ao usuário.	Média
RNF7	O sistema deve utilizar um sistema de autenticação como o Amazon Cognito.	Média

RNF8	O sistema deve ter portabilidade para os principais navegadores.	Alta
RNF9	O sistema deve ser responsivo para os diversos tipos de tela atuais.	Alta
RNF10	O sistema deve ser desenvolvido com banco de dados relacional para persistência de dados.	Alta

Tabela 2. Requisitos Não Funcionais

1.2 Metas e Qualidade

Conteúdo

Os três principais objetivos de qualidade para a arquitetura da plataforma "Take-A-Guide" são essenciais para garantir a satisfação dos stakeholders principais. Estes objetivos se concentram em aspectos específicos de qualidade arquitetônica, baseando-se em tópicos do padrão ISO 25010. Abaixo estão listados os objetivos de qualidade com seus respectivos cenários concretos, ordenados por prioridade.

Objetivo de Qualidade	Descrição	Cenário Concreto
Segurança	Garantir a proteção dos dados dos usuários e a segurança nas transações financeiras realizadas na plataforma.	Implementar criptografia robusta para armazenamento de dados e durante a transmissão. Utilizar autenticação multifator para acesso de administradores e validação de identidade em transações de alto valor.
Escalabilidade	Manter a performance eficiente mesmo com o crescimento no número de usuários e nas operações realizadas.	Utilizar serviços de computação em nuvem para balanceamento de carga e escalabilidade automática, garantindo que a plataforma suporte milhares de acessos simultâneos sem

		degradação de desempenho.
Usabilidade	Proporcionar uma experiência de usuário intuitiva e fácil de usar, tanto para turistas quanto para guias.	Realizar testes de usabilidade com usuários finais para validar o design da interface. Garantir que todas as funcionalidades sejam acessíveis em poucos cliques e que o processo de busca, reserva e pagamento seja direto e simples.

Tabela 4. Objetivo de Qualidade

Motivação

Conhecer os objetivos de qualidade dos principais stakeholders é crucial, pois eles influenciam diretamente as decisões arquitetônicas fundamentais. Ao priorizar a segurança, escalabilidade e usabilidade, a plataforma "Take-A-Guide" assegura que atende às expectativas dos turistas, guias e administradores. Ser concreto sobre esses objetivos evita o uso de palavras vazias e garante que as necessidades dos stakeholders sejam realmente atendidas.

1.3 Stakeholders

Conteúdo

Esta seção fornece uma visão geral explícita dos stakeholders do sistema "Take-A-Guide". Estes são todos os indivíduos, funções ou organizações que precisam conhecer a arquitetura, ser convencidos de sua eficácia, trabalhar com a arquitetura ou o código, necessitar da documentação da arquitetura para suas atividades, ou que precisam tomar decisões sobre o sistema ou seu desenvolvimento.

Motivação

É crucial conhecer todas as partes envolvidas no desenvolvimento do sistema ou que são afetadas por ele. Sem essa compreensão, podem surgir surpresas indesejadas durante o processo de desenvolvimento. Esses stakeholders determinam a extensão e o nível de detalhamento do trabalho e dos resultados esperados.

Consultores Jurídicos	**Contato**	**Expectativas**

Turista (Usuários Finais)	N/A	Experiência de usuário intuitiva, segura e personalizada. Proteção de dados pessoais e facilidade de uso.
Guia Turístico	N/A	Interface fácil de usar para criação e gerenciamento de perfis, confiabilidade no agendamento de tours e recebimento de pagamentos.
Administrador da Plataforma	N/A	Ferramentas de administração eficientes, monitoramento de desempenho do sistema e satisfação dos usuários.

Desenvolvedor de Software	N/A	Documentação técnica elara, arquitetura modular e bem definida, padrões de segurança e boas práticas de codificação.
Arquiteto de Software	N/A	Diretrizes arquitetônicas elaras, suporte para escala bilidade, segurança e alta disponibilidade.
Equipe de Segurança de TI	N/A	Arquitetura que suporte criptografia, autenticação multifator, e conformidade com regulamentações de segurança.
Investidor/Patrocinador	N/A	Garantia de viabilidade técnica, potencial de crescimento, e retorno sobre investimento.

Equipe de Marketing e Vendas	N/A	Integração com ferramentas de análise de métricas, suporte para campanhas de marketing e estratégias de aquisição de usuários.
Consultores Jurídicos	N/A	Conformidade com leis e regulamentações de proteção de dados, políticas de privacidade bem definidas.

Tabela 5. Stakeholders

2. Restrições de Arquitetura

As poucas restrições deste projeto estão refletidas na solução final. Esta seção as apresenta e, se aplicável, sua motivação.

2.1 Restrições técnicas

ID	Restrições	Background e/ou motivação
	Restrições de Software e Programação	
1	Implementação em Java	A aplicação deve integrar-se ao Java 8 e Spring Boot, que são tecnologias amplamente utilizadas e reconhecidas por sua robustez e eficiência no desenvolvimento de aplicações. Entretanto, a interface da aplicação, ou seja, a API, deve ser projetada para ser independente de linguagem e framework. Isso significa que deve ser desenvolvida com padrões e práticas que permitam sua utilização por diversos clientes, independentemente da tecnologia utilizada em sua implementação.
2	Compatibilidade com ambientes de nuvem	A aplicação deve ser projetada para ser facilmente implantada em ambientes de nuvem, como Azure, garantindo escalabilidade e flexibilidade.
3	Documentação da API	A API deve ser acompanhada de uma documentação clara e acessível, utilizando ferramentas como Swagger, para facilitar a compreensão e uso por desenvolvedores externos.
4	Suporte a formatos JSON e XML	A API deve suportar tanto JSON quanto XML para atender diferentes necessidades dos clientes e facilitar a interoperabilidade entre sistemas.
5	Persistência em banco de dados SQL Azure	A aplicação deve utilizar o Azure SQL Database como solução de persistência. Essa escolha garante que a aplicação aproveite os recursos de escalabilidade, alta disponibilidade e segurança oferecidos pela plataforma de

		nuvem Azure. O uso de um banco de dados relacional também permite que a aplicação mantenha a integridade dos dados e suporte transações complexas. Além disso, a integração com outras ferramentas e serviços do Azure facilita a implementação de soluções mais robustas e escaláveis.
6	Implementação em TypeScript	A escolha do TypeScript como linguagem de programação visa proporcionar tipagem estática, o que melhora a detecção de erros durante o desenvolvimento e facilita a manutenção do código. Além disso, sua compatibilidade com JavaScript permite a utilização de bibliotecas existentes, enquanto promove boas práticas de desenvolvimento, como a modularidade e a organização do código.
Restrições de Sistema Operacional		
1	Desenvolvimento independente de OS	A aplicação deve ser compilável em todos os três principais sistemas operacionais (Mac OS X, Linux e Windows). Essa abordagem garante que o software seja acessível a uma base de usuários diversificada, permitindo que desenvolvedores e usuários finais possam utilizá-lo independentemente de sua plataforma preferida.
Restrições de Hardware		
1	Eficiente em memória	Cada megabyte de memória utilizado em uma solução em nuvem acarreta custos adicionais, portanto, a eficiência no uso da memória é crucial para manter a viabilidade econômica da aplicação.
2	Requisitos Mínimos de CPU	A aplicação deve ser projetada para rodar em processadores com especificações mínimas, garantindo que funcione adequadamente em uma variedade de ambientes, incluindo máquinas de baixo custo e servidores em nuvem. Isso aumenta a acessibilidade e reduz barreiras para a implementação.
3	Escalabilidade Vertical	Escalabilidade vertical permite que a aplicação se adapte a aumentos de carga sem a necessidade de reestruturar a arquitetura ou a lógica de distribuição de carga.

4	Latência mínima em comunicação	A aplicação deve ser projetada para minimizar a latência em comunicação com outros serviços e bancos de dados. Isso é essencial para garantir um desempenho aceitável, especialmente em ambientes de nuvem, onde a latência pode impactar a experiência do usuário.

Tabela 6. Restrições Técnicas

2.2 Restrições Organizacionais

ID	Restrições	Background e/ou motivação
1	Equipe- Responsabilidades	Bruno Marques (Back-End), João Elias Fadel (Devops) João Victor (Tech Lead), José Mussy (Documentação) e Lohine Mussi (Documentação).
2	Cronograma	Início em agosto de 2024, com o desenvolvimento da ideia e a coleta dos requisitos. Em seguida, iniciamos a elaboração do modelo C4, a documentação segundo o Arc42 e a prototipação dos diagramas. Paralelamente, trabalhamos de forma colaborativa no desenvolvimento do back-end, na configuração do banco de dados e na implementação do front-end, garantindo uma integração fluida entre as diferentes camadas da aplicação.
3	Configuração e Controle de Versão / Gestão	Repositório Git privado com um histórico completo de commits e uma branch principal pública enviada para o GitHub, vinculada ao projeto.
4	Comunicação e colaboração	Um canal de comunicação estabelecido (Discord) para facilitar a colaboração entre os membros da equipe e garantir que as atualizações e decisões sejam compartilhadas de forma eficaz.

Tabela 7. Restrições Organizacionais

2.3 Convenções

ID	Restrições	Background e/ou motivação
1	Documentação da Arquitetura	Estrutura baseada no template arc42 em inglês, na versão 8.2.

2	Idioma	O projeto correspondente tem como alvo um público internacional, portanto, o inglês deve ser utilizado em todo o projeto.
3	Convenções de Codificação	O projeto utiliza as Convenções de Código para Java e Spring Boot. As convenções são aplicadas por meio do Checkstyle.
4	Controle de Versão	O código deve ser gerenciado em um repositório Git privado, garantindo que todo o histórico de commits seja mantido e que haja uma branch principal pública para colaboração.
5	Compatibilidade Multiplataforma	A aplicação deve ser projetada para rodar em diferentes sistemas operacionais (Windows, Linux, Mac OS), garantindo que funcione de forma consistente em ambientes diversos.

Tabela 8. Convençoes

3. Escopo do Sistema e Contexto

3.1 Contexto de negócio

O sistema “Take-a-Guide” conecta turistas a guias turísticos através de uma plataforma online, permitindo a personalização e reserva de tours. O objetivo principal é facilitar a experiência de viagem dos turistas e aumentar a visibilidade dos guias, democratizando o acesso ao turismo de forma personalizada.

Atores externos:

- **Turista:** Usuário que utiliza a plataforma para buscar guias e personalizar seus passeios.
- **Guia turístico:** Profissional que oferece serviços de turismo, gerencia perfis e disponibiliza passeios na plataforma.
- **Administrador:** usuário que tem acesso aos dados operacionais, monitorando o registro de guias turísticos na plataforma, aprovando ou não a documentação necessária do Know Yout Customer (KYC).

Interações principais:

- O turista utiliza a plataforma para consultar perfis de guias e reservar tours personalizados.
- O guia turístico utiliza a plataforma para cadastrar seus passeios, gerenciar seus passeios personalizados e visualizar reservas.
- O administrador utiliza ferramenta interna para monitorar a etapa de aprovação KYC do usuário padrão para se tornar um guia afiliado a plataforma.

3.2 Contexto técnico

O sistema "Take-A-Guide" integra-se com diversos serviços externos para fornecer suas funcionalidades principais. Entre eles, estão serviços de autenticação, mapas e processamento

de pagamentos. Essas integrações são essenciais para garantir uma experiência de usuário segura e fluida.

Sistemas Externos:

- **Google Maps:** Utilizado para exibir mapas e rotas relevantes para os turistas e guias.
- **Sistema de Pagamentos (Mercado Pago):** Responsável por processar as transações financeiras entre turistas e guias.
- **Sistema de Autenticação (OAuth via Google, Apple, Microsoft):** Utilizado para autenticação de usuários (turistas, guias e administradores).

3.3 Limitações do Escopo

O sistema "Take-A-Guide" limita-se a fornecer uma plataforma de intermediação entre turistas e guias, sem realizar ou se responsabilizar diretamente pelos passeios. O sistema também não lida diretamente com transações financeiras, delegando essa função para um serviço externo (Mercado Pago). Além disso, o sistema não se responsabiliza por falhas nos serviços externos de autenticação e mapas.

4. Estratégia da Solução

Esta seção resume as decisões fundamentais e estratégias que moldam a arquitetura do sistema da plataforma Take-A-Guide, incluindo:

Decisões Tecnológicas

- Grafana e Prometheus: Prometheus coleta e agrega métricas, enquanto o Grafana cria dashboards para insights e análises. A integração dessas ferramentas ajuda o Take-A-Guide a se tornar uma plataforma mais robusta, confiável e orientada a dados, permitindo que a equipe gerencie e otimize a operação de maneira contínua e eficiente.
- Kubernetes: Orquestração de containers que automatiza o gerenciamento, a implantação e a escalabilidade de aplicações conteinerizadas. Kubernetes permite que as aplicações sejam executadas de maneira eficiente, distribuída e resiliente, gerenciando clusters de contêineres em grande escala. Isso oferece uma experiência de usuário mais estável e responsiva, otimiza o uso de recursos e simplifica a gestão operacional, sendo crucial para uma plataforma em expansão.
- MongoDB: O MongoDB foi escolhido por sua capacidade de lidar eficientemente com grandes volumes de dados não estruturados e semi-estruturados, graças ao seu modelo de dados flexível baseado em documentos. Para a plataforma Take-A-Guide, que precisa armazenar informações diversas e em constante mudança sobre turistas, guias, rotas e preferências, o MongoDB proporciona agilidade na manipulação de dados sem a necessidade de esquemas rígidos. Sua arquitetura distribuída garante alta disponibilidade e escalabilidade horizontal, atendendo à demanda de crescimento dinâmico da plataforma. A flexibilidade do MongoDB também facilita o gerenciamento de dados geoespaciais, um requisito essencial para a personalização dos passeios com base em localizações.
- SQL Server Azure: O SQL Server Azure foi integrado à arquitetura do Take-A-Guide para gerenciar dados transacionais críticos e garantir consistência e integridade em operações que exigem um alto nível de controle, como agendamento de tours, pagamentos e autenticações de usuários. Por ser um banco de dados relacional baseado em cloud, ele

oferece a robustez e a segurança de que a plataforma precisa, além de escalabilidade vertical que suporta cargas de trabalho intensivas sem comprometer a performance. A escolha do SQL Server Azure também se alinha às necessidades de conformidade com normas de segurança, como a LGPD, uma vez que oferece criptografia avançada e backups automatizados, garantindo proteção e recuperação de dados.

- Keptn: Ferramenta de automação para CI/CD (Continuous Integration/Continuous Delivery) que automatiza e orquestra pipelines de entrega de software. Keptn garante a qualidade e a estabilidade das implantações, assegurando que cada atualização seja implementada de forma controlada e com o mínimo de risco.

Decomposição do Sistema em Alto Nível:

- Arquitetura de Monitoramento e Orquestração: A integração de ferramentas como Prometheus e Grafana para monitoramento, e Kubernetes para orquestração de containers, representa padrões arquiteturais importantes para a eficiência e resiliência do sistema.
- Banco de Dados e CI/CD: A escolha de Azure SQL Server para gerenciamento de dados e Keptn para automação de CI/CD são decisões estratégicas que suportam a escalabilidade e a qualidade contínua das atualizações do sistema.

Alcance das Metas de Qualidade:

- Eficiência Operacional e Resiliência: O uso combinado de Kubernetes, Azure SQL Server e Gradle assegura operações eficientes e resilientes, enquanto a integração de Prometheus e Grafana contribui para uma gestão de desempenho baseada em dados. Keptn ajuda a manter a qualidade das implantações com mínima interrupção.

Decisões Organizacionais:

- Processos de Desenvolvimento e Gestão: A escolha dessas tecnologias e ferramentas está alinhada com a estratégia de desenvolvimento para assegurar uma plataforma estável, escalável e responsiva, otimizar a gestão de recursos e automatizar processos críticos.

Motivação

Essas decisões foram tomadas para garantir que a plataforma Take-A-Guide seja robusta, confiável e escalável, atendendo às necessidades de crescimento e desempenho contínuo. Cada ferramenta e tecnologia foi selecionada para apoiar a eficiência operacional, a gestão de dados, e a automação de processos, alinhando-se com os objetivos de qualidade e as restrições do projeto.

Forma

As explicações para as escolhas tecnológicas são mantidas breves e focadas na motivação por trás de cada decisão, com base na declaração do problema, metas de qualidade e restrições principais.

5. Visualização de bloco de construção

5.1 Level 1

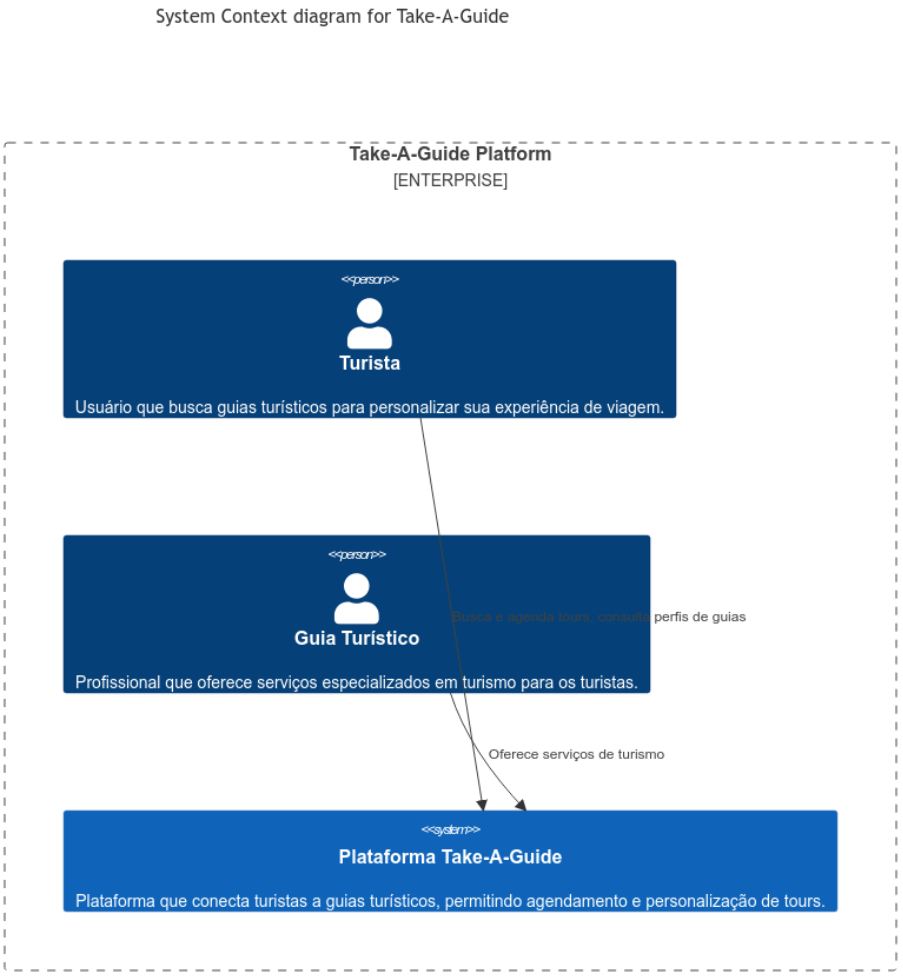

Figura 1. C4 model Level 1

```
C4Context

title System Context diagram for Take-A-Guide

Enterprise_Boundary(b0, "Take-A-Guide Platform") {

Person(turista, "Turista", "Usuário que busca guias turísticos para personalizar sua experiência de viagem.")

Person(guia, "Guia Turístico", "Profissional que oferece serviços especializados em turismo para os turistas.")



System(Plataforma, "Plataforma Take-A-Guide", "Plataforma que conecta turistas a guias turísticos, permitindo agendamento e personalização de tours.")

}

Rel(turista, Plataforma, "Busca e agenda tours, consulta perfis de guias")
Rel(guia, Plataforma, "Oferece serviços de turismo")
```

5.1.2 Descrição

Interfaces:

- **Google Maps:** Fornece serviços de localização e exibição de mapas, incluindo pontos turísticos e rotas.
- **Sistema de Pagamentos:** Processa transações financeiras para reservas de passeios.
- **Sistema de Autenticação:** Gerencia a autenticação de usuários por meio de contas do Gmail, Outlook ou Apple.

Características de Desempenho/Qualidade:

- **Disponibilidade:** Alta disponibilidade para garantir que os usuários possam acessar a plataforma a qualquer momento.
- **Desempenho:** Resposta rápida para consultas e processamento de pagamentos, assegurando uma experiência de usuário fluida e eficiente.

Requisitos:

- **Integração com Sistemas Externos:** A plataforma deve integrar-se com serviços externos para autenticação e processamento de pagamentos.
- **Funcionalidades:** Inclusão de funcionalidades para busca e agendamento de passeios.
- **Riscos:** Segurança de Dados: Garantir a proteção e segurança dos dados dos usuários durante o processo de autenticação e processamento de pagamentos.

Dependências Importantes:
Sistema de Autenticação

- **Propósito/Responsabilidade:** Gerencia a autenticação de usuários através de serviços de login externos.
- **Sintaxe/Semântica:** APIs de autenticação para Gmail, Outlook

  e Apple. Protocolos: OAuth 2.0 para autenticação.
- **Tratamento de Erros**: Respostas de erro para falhas de autenticação ou credenciais inválidas.
- **Versões:** Versão atual das APIs de autenticação.

- **Qualidades:** Segurança robusta e conformidade com padrões de autenticação.

5.2 Level 2

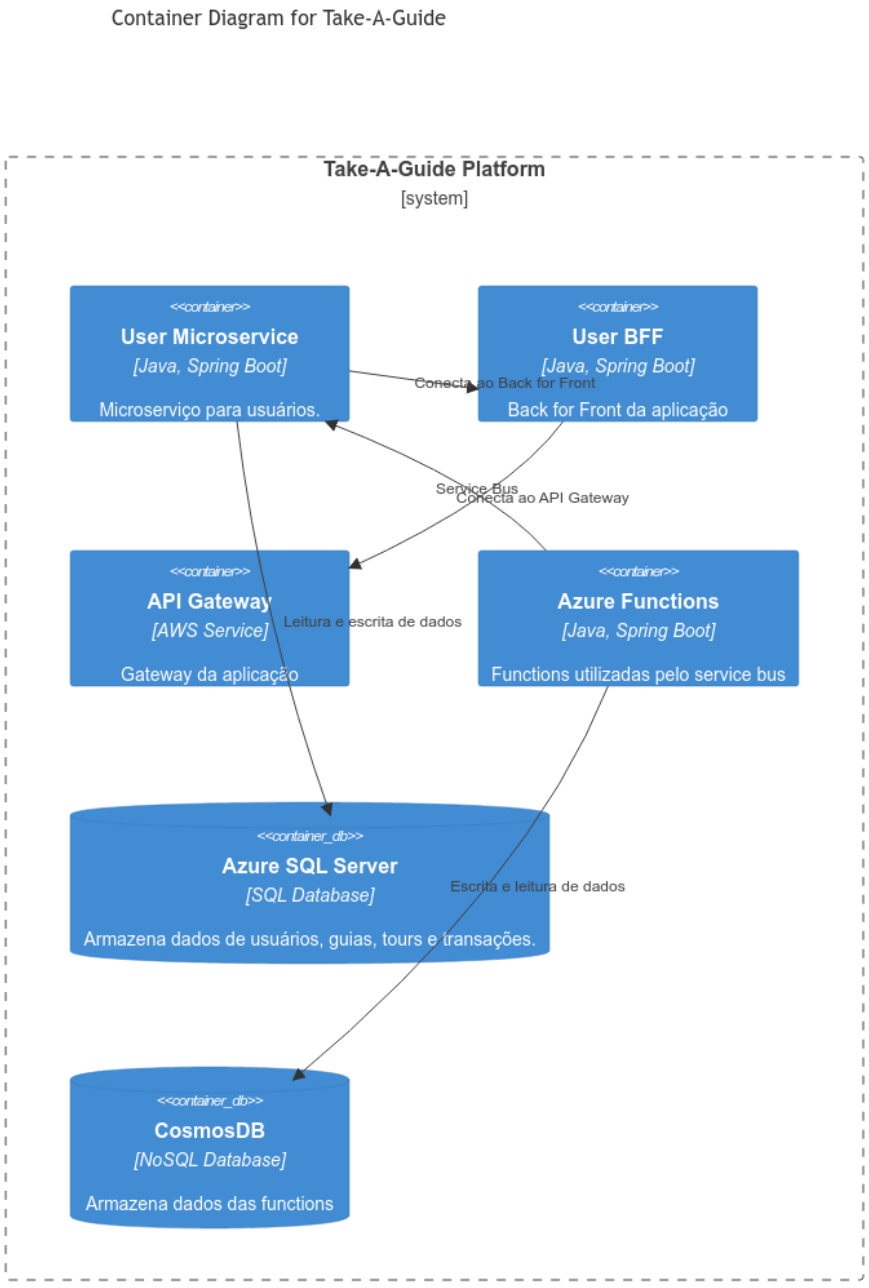

Figura 2. C4 model Level 2

```
C4Container

    title Container Diagram for Take-A-Guide


    Boundary(b0, "Take-A-Guide Platform") {

          Container(UserService,"User Microservice", "Java,
Spring Boot", "Microserviço para usuários.")

           Container(bff, "User BFF", "Java, Spring Boot",
"Back for Front da aplicação")

              Container(gate, "API Gateway","AWS Service",
"Gateway da aplicação")

          Container(azure, "Azure Functions", "Java, Spring
Boot", "Functions utilizadas pelo service bus")

            ContainerDb(Database, "Azure SQL Server", "SQL
Database", "Armazena dados de usuários, guias, tours e
transações.")

          ContainerDb(Cosmos, "CosmosDB", "NoSQL Database",
"Armazena dados das functions")


    }

    Rel(UserService, bff, "Conecta ao Back for Front")

    Rel(azure, Cosmos, "Escrita e leitura de dados")

    Rel(azure,UserService, "Service Bus")
```

```
    Rel(bff, gate, "Conecta ao API Gateway")

        Rel(UserService, Database, "Leitura e escrita de
dados")
```

5.2.1 Descrição

Contêineres Aplicativo Monolítico
Tecnologia:Java, Spring Boot

Descrição: Plataforma central que conecta turistas a guias turísticos, permitindo o agendamento e a personalização de passeios. Gerencia todas as operações principais, incluindo o gerenciamento de usuários, guias e reservas.

Banco de Dados

Tecnologia: Azure SQL Server

Descrição: Banco de dados SQL que armazena informações críticas sobre usuários, guias, passeios e transações financeiras. Garante integridade e disponibilidade dos dados, aproveitando a escalabilidade e segurança do Azure.

5.3 Level 3

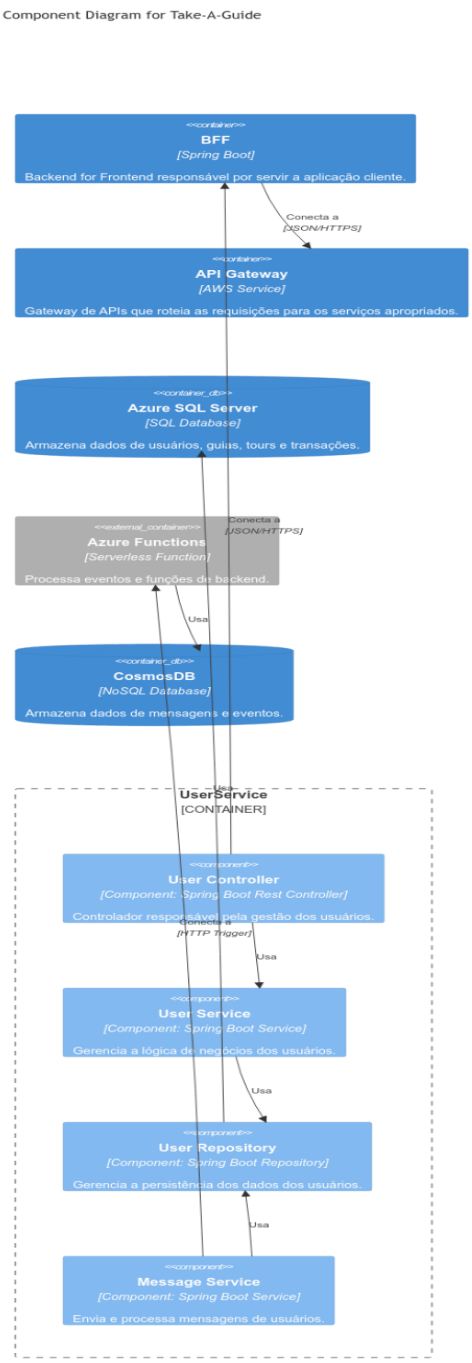

Figura 3. C4 model Level 3

```
C4Component
    title Component Diagram for Take-A-Guide

    Container_Boundary(takeAGuideApp, "UserService") {

        Component(userController, "User Controller", "Component: Spring
Boot Rest Controller", "Controlador responsável pela gestão dos
usuários.")
        Component(userService, "User Service", "Component: Spring Boot
Service", "Gerencia a lógica de negócios dos usuários.")
        Component(userRepository, "User Repository", "Component: Spring
Boot Repository", "Gerencia a persistência dos dados dos usuários.")
        Component(messageService, "Message Service", "Component: Spring
Boot Service", "Envia e processa mensagens de usuários.")
    }

    Container(bff, "BFF", "Spring Boot", "Backend for Frontend
responsável por servir a aplicação cliente.")
    Container(apiGateway, "API Gateway", "AWS Service", "Gateway de APIs
que roteia as requisições para os serviços apropriados.")
    ContainerDb(Azure, "Azure SQL Server", "SQL Database", "Armazena
dados de usuários, guias, tours e transações.")

    Container_Ext(azureFunctions, "Azure Functions", "Serverless
Function", "Processa eventos e funções de backend.")
    ContainerDb(CosmosDB, "CosmosDB", "NoSQL Database", "Armazena dados
de mensagens e eventos.")

    Rel(userController, userService, "Usa")
    Rel(userService, userRepository, "Usa")
    Rel(userRepository, Azure, "Usa")
    Rel(userController, bff, "Conecta a", "JSON/HTTPS")
    Rel(bff, apiGateway, "Conecta a", "JSON/HTTPS")
    Rel(messageService, userRepository, "Usa")
    Rel(messageService, azureFunctions, "Conecta a", "HTTP Trigger")
    Rel(azureFunctions, CosmosDB, "Usa")
```

5.4 Level 4

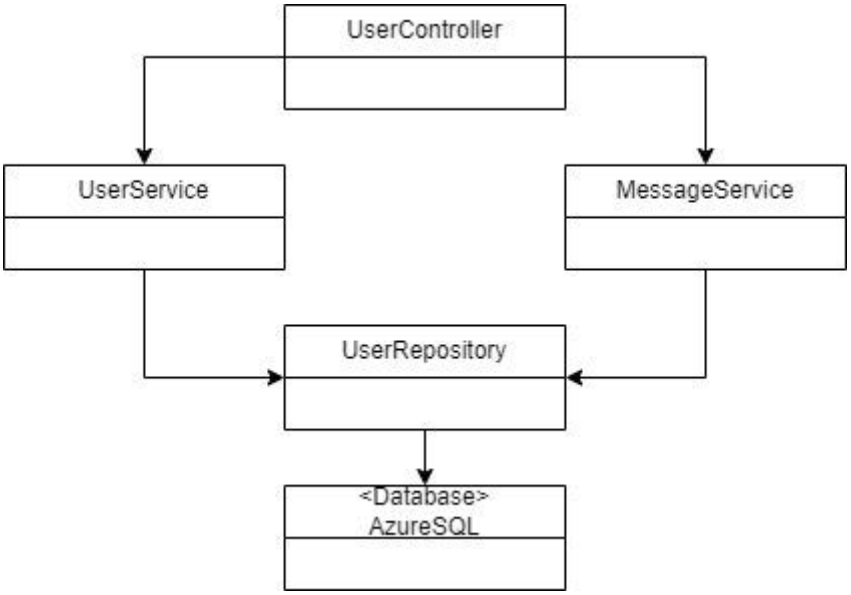

Figura 4. C4 model Level 4

classDiagram

```
        class UserController
        class UserService
        class MessageService
        class UserRepository


    class AzureSQL {
        <<Database>>
    }

    UserController --> UserService : "depende"
    UserService --> UserRepository
    MessageService --> UserRepository
```

6. Visão de Runtime

Para a plataforma "Take-A-Guide", essa seção foca em ilustrar o comportamento dinâmico do sistema durante os principais fluxos de uso, como autenticação de usuários, agendamento de passeios e processamento de pagamentos. A seguir, são descritos alguns cenários de runtime para representar essas interações.

6.1 Cenário 1: Autenticação de Usuário

Descrição: Este cenário descreve o processo de login de um usuário (turista ou guia) na plataforma usando um provedor de autenticação externo (Google, Apple, Microsoft).
Passos:

1. O usuário acessa a página de login no front-end (Next.js).
2. O front-end envia uma solicitação ao serviço de autenticação da plataforma (Node.js Express), solicitando uma sessão de login.
3. O serviço de autenticação redireciona o usuário para a página de autenticação externa (Google, Apple, Microsoft).
4. O usuário insere suas credenciais no provedor externo.
5. O provedor externo autentica o usuário e retorna um token de autenticação para o serviço de autenticação da plataforma.
6. O serviço de autenticação valida o token e cria uma sessão para o usuário, retornando um token de sessão para o front-end.
7. O front-end armazena o token e redireciona o usuário para a página principal.

Aspectos Notáveis:

- Segurança: A utilização de OAuth 2.0 garante um processo de login seguro e em conformidade com as regulamentações de proteção de dados.
- Tempo de Resposta: O tempo de resposta é um fator crítico neste cenário, sendo necessário garantir que o processo de autenticação seja rápido para proporcionar uma boa experiência ao usuário.

6.2 Cenário 2: Reserva de Tour

Descrição: Este cenário descreve como um turista agenda um tour com um guia utilizando a plataforma.
Passos:

1. O turista faz login na plataforma.
2. O turista navega até a pesquisa de tours disponíveis e filtra por localização, especialidade ou avaliações de outros usuários.

3. O turista seleciona um tour disponível e visualiza suas informações, incluindo vagas, datas e horários disponíveis.
4. O turista escolhe uma data e horário e clica em "Reservar Tour".
5. O sistema envia uma solicitação ao back-end (Java Spring Boot), que valida a disponibilidade do guia.
6. O back-end confirma a reserva e armazena os detalhes no banco de dados.
7. O sistema abre o cenário de pagamento.
8. O turista realiza o pagamento e a plataforma retorna status de sucesso.
9. O sistema envia um e-mail de notificação para o guia, informando sobre a nova reserva.
10. O sistema envia um e-mail de notificação para o turista, informando que a reserva foi realizada com sucesso eos dados da reserva.
11. O front-end exibe a confirmação da reserva para o turista.

Aspectos Notáveis:

- Confiabilidade: É essencial garantir que o sistema de agendamento seja confiável e evite overbooking (duplicação de reservas).
- Notificações: O sistema envia notificações por e-mail para ambas as partes, garantindo que o guia esteja ciente do novo agendamento.

6.3 Cenário 3: Processamento de Pagamento

Descrição: Este cenário ilustra o fluxo de pagamento entre o turista e o guia por meio do serviço externo Mercado Pago.

Passos:

1. O turista seleciona um passeio e agenda um tour.
2. Após a confirmação do agendamento, o sistema solicita que o turista efetue o pagamento.
3. O turista insere seus dados de pagamento no front-end.
4. O sistema envia os detalhes de pagamento ao serviço de pagamento (Mercado Pago) por meio de uma API.
5. O Mercado Pago processa o pagamento e retorna um status de sucesso ou falha.
6. Se o pagamento for bem-sucedido, o sistema confirma a reserva e atualiza o banco de dados.
7. O sistema envia e-mails de confirmação para o turista e o guia.
8. Se o pagamento falhar, o sistema notifica o turista e oferece a opção de tentar novamente.

Aspectos Notáveis:

- Segurança: A integração com o Mercado Pago garante a segurança das transações financeiras, utilizando HTTPS e criptografia de ponta a ponta.
- Recuperação de Erros: O sistema lida com falhas de pagamento de forma eficiente,

notificando o usuário e oferecendo opções para tentar novamente.

6.4 Cenário 4: Registro de Novo Anúncio pelo Guia

Descrição: Este cenário descreve o processo pelo qual um guia pode criar e publicar um novo anúncio de tour na plataforma.

Passos:

1. O guia faz login na plataforma através de um provedor de autenticação externo (Google, Apple, Microsoft), seguindo o processo descrito no cenário 1.
2. O guia navega até o dashboard de guia, onde pode visualizar e gerenciar seus anúncios e reservas atuais.
3. O guia clica no botão "Criar Novo Anúncio" e é redirecionado para a página de criação de anúncio.
4. O sistema exibe um formulário onde o guia deve preencher os seguintes campos obrigatórios:
    - Nome do tour.
    - Descrição do tour.
    - Localização (cidade e pontos de interesse cobertos).
    - Número máximo de participantes.
    - Data e horários disponíveis.
    - Preço por pessoa.
    - Categoria (histórico, cultural, aventura, etc.).
5. O guia preenche as informações e clica em "Salvar".
6. O sistema valida os dados inseridos:
    - Verifica se todos os campos obrigatórios estão preenchidos.
    - Checa a disponibilidade de datas e horários, para evitar conflitos com outros anúncios do guia.
7. O sistema confirma a criação do novo anúncio e o salva no banco de dados.
8. O guia recebe uma notificação por e-mail confirmando o registro do novo anúncio.
9. O anúncio é publicado automaticamente e torna-se visível para turistas na plataforma.
10. O guia pode gerenciar o novo anúncio no dashboard, incluindo a opção de editá-lo ou pausá-lo.

Aspectos Notáveis:

- Validação de Dados: O sistema realiza validação rigorosa para garantir que as informações fornecidas são precisas e que não há sobreposição de datas/horários.
- Notificações: O sistema garante que o guia receba uma confirmação por e-mail quando o anúncio for publicado, garantindo clareza no processo.
- Experiência do Usuário: O processo de criação do anúncio deve ser simples e

intuitivo, maximizando a eficiência do guia na plataforma.

7. Visão de Implantação

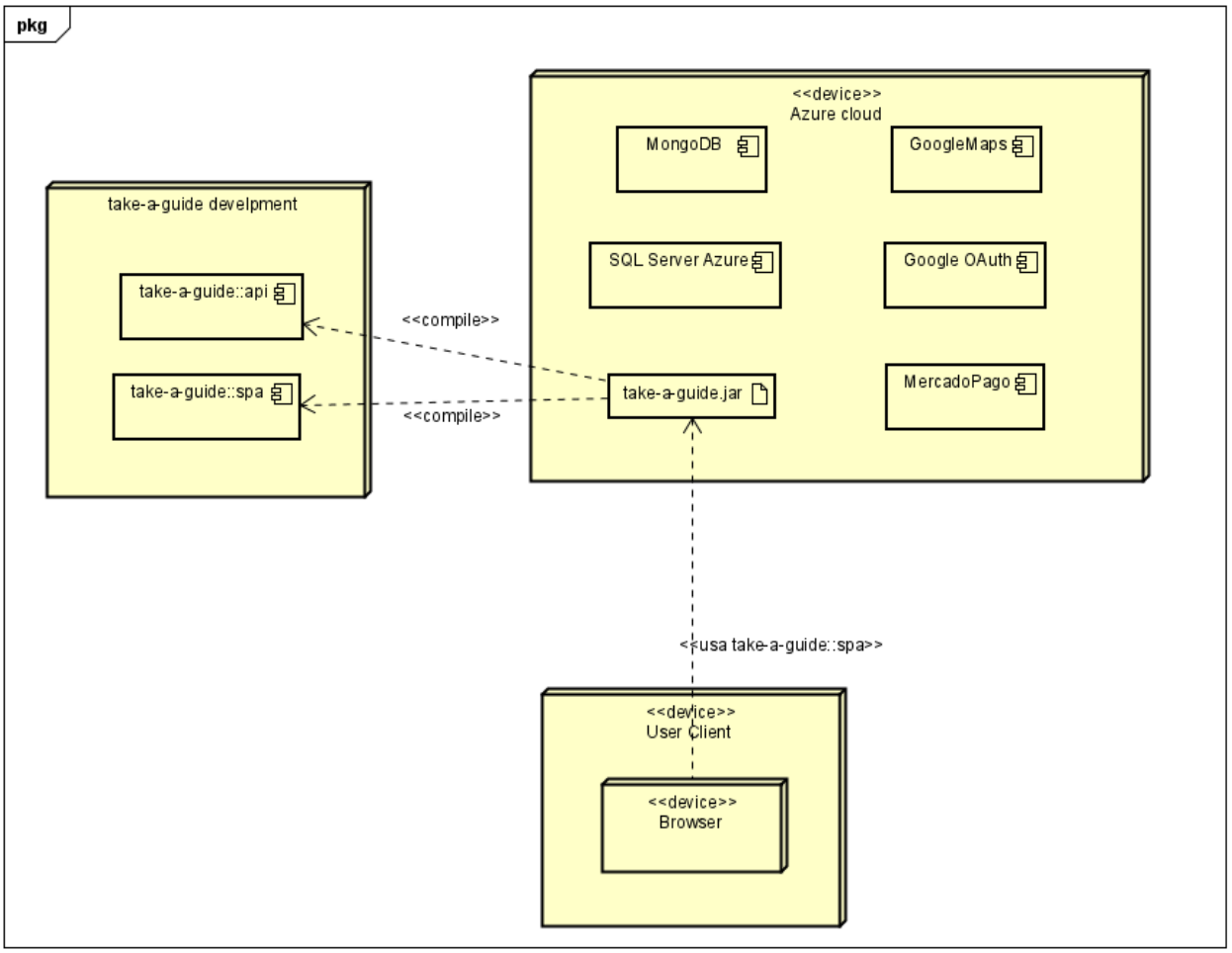

Figura 5 – Diagrama de implantação.

Node / Componente	Descrição
take-a-guide development	Ambiente de desenvolvimento com Next.js, Java Spring Boot, Azure SQL Server/MongoDB para testes.
Cloud host (AWS/Azure)	Servidor cloud que hospeda o sistema. Backend e frontend são orquestrados por Kubernetes.

Kubernetes Cluster	Orquestra containers para frontend (Next.js) e backend (Spring Boot), com escalabilidade automática.
Azure SQL Server/MongoDB	Bancos de dados distribuídos, Azure SQL Server para dados transacionais e MongoDB para dados não estruturados.
OAuth (Google, Apple)	Serviço externo de autenticação, utilizado para o login dos usuários (turistas, guias, admins).
Mercado Pago	Serviço externo de pagamento, responsável por processar transações financeiras de forma segura.
Take-a-guide.jar	Um “fat jar” que contém todas as dependências Java necessárias para o backend Spring Boot. O arquivo jar pode ser executado em containers Docker.
GoogleMaps	Serviço externo Utilizado para exibir mapas e rotas relevantes para os turistas e guias.
Navegador (Browser)	Interface utilizada pelos usuários para acessar o Take-A-Guide. Compatível com Chrome, Firefox, Edge.

Tabela 9 – Nodes e Componentes da implantação.

8. Conceitos Transversais

8.1 Modelo de domínio

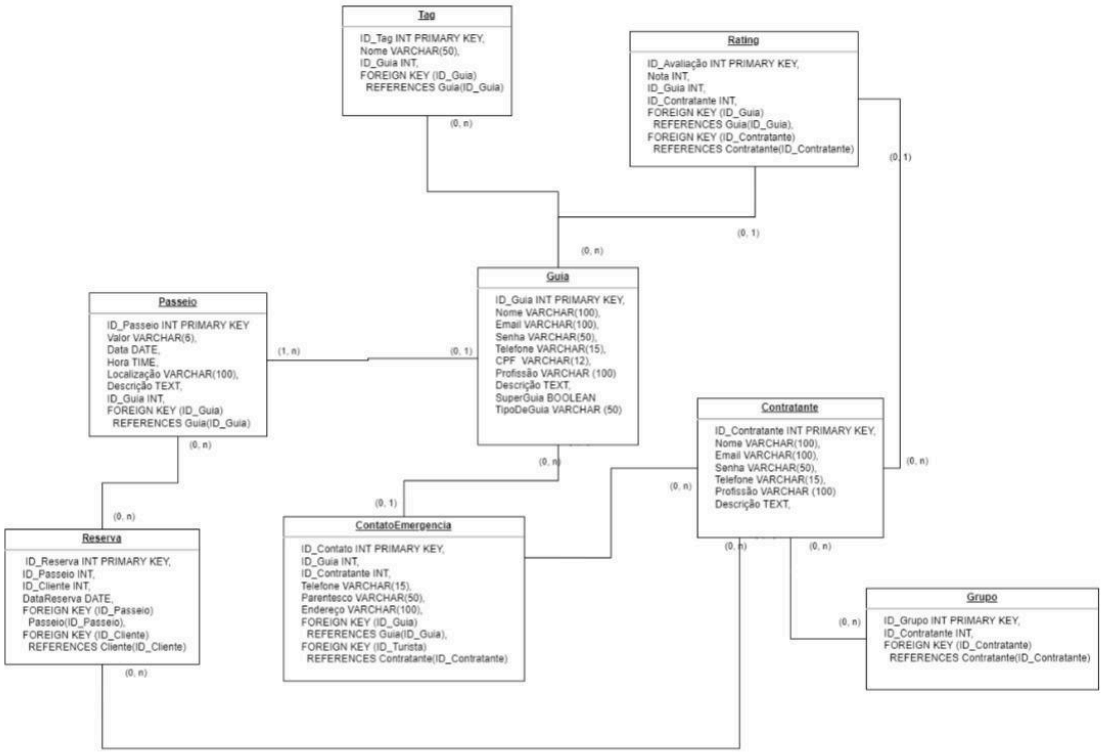

Figura 6 – Diagrama de Entidade Relacionamento (DER).

Nome	Descrição
Guia	Armazena informações sobre os guias turísticos. Contém dados como nome, email, telefone, CPF, especialização e tipo de guia. Um guia pode oferecer vários passeios.
Contratante	Armazena informações sobre o contratante (turista). Contém dados pessoais como nome, email, senha, telefone e profissão. Um contratante pode reservar vários passeios.
Passeio	Contém informações sobre os passeios ofertados pelos guias, incluindo valor, data, horário, localização e descrição. Cada passeio é associado a um guia.
Reserva	Armazena dados sobre as reservas feitas pelos contratantes para os passeios. Inclui informações como o ID do passeio, valor e a data de reserva.

Rating	Armazena as avaliações feitas pelos contratantes para os guias. Contém uma nota de avaliação e é relacionado tanto ao guia quanto ao contratante.
Tag	Armazena as tags associadas aos guias, como especializações ou áreas de atuação. Cada tag está associada a um guia.
ContatoEmergencia	Armazena informações de contato de emergência de guias ou contratantes. Cada contato está relacionado a um guia e/ou a um contratante.
Grupo	Armazena os dados de grupos que podem ser formados por contratantes. Cada grupo está associado a um ou mais

Tabela 10 – Descrição das Entidades Relacionamentos.

9. Decisões Arquiteturais

A seguir, estão as principais decisões arquiteturais tomadas para a plataforma "Take-A-Guide" e suas justificativas.

9.1 Decisão 1: Uso do Framework Next.js para o Front-end

Decisão: O framework Next.js foi escolhido para o desenvolvimento do front-end da plataforma "Take-A-Guide".

Motivação:

- Performance: O Next.js oferece renderização do lado do servidor (SSR), melhorando a velocidade de carregamento das páginas e a performance geral da aplicação, o que é essencial para uma plataforma que precisa oferecer uma boa experiência ao usuário.
- SEO otimizado: A renderização SSR também contribui para otimizar o SEO, facilitando que a plataforma seja encontrada em motores de busca, aumentando a visibilidade para turistas e guias.
- Facilidade de uso: O Next.js é baseado no React, um framework amplamente utilizado, garantindo uma comunidade ativa e suporte robusto.

Consequências:

- A aplicação terá uma boa performance em múltiplos dispositivos, além de ser fácil de manter e escalar conforme necessário.

9.2 Decisão 2: Uso do Spring Boot para o Back-end

Decisão: O Spring Boot foi escolhido como o framework principal para o desenvolvimento do back-end.

Motivação:

- Escalabilidade e Robustez: Spring Boot é um framework consolidado, amplamente utilizado para o desenvolvimento de aplicações empresariais escaláveis e robustas.
- Integração com APIs: O Spring Boot oferece suporte nativo para integração com diversas APIs e sistemas externos, como serviços de pagamento (Mercado Pago) e autenticação (OAuth).
- Comunidade e Suporte: A vasta comunidade de desenvolvedores de Spring Boot garante a disponibilidade de documentação e boas práticas, além de fornecer um ecossistema maduro para desenvolvimento de aplicações web.

Consequências:

- A escolha do Spring Boot garante uma base sólida para o back-end da plataforma, que pode ser facilmente expandida para suportar novas funcionalidades no futuro.

9.3 Decisão 3: Adoção de Kubernetes para Orquestração de Containers

Decisão: A plataforma será implantada utilizando Kubernetes para orquestração de containers.

Motivação:

- Escalabilidade: Kubernetes permite escalar os containers horizontalmente conforme o número de usuários cresce, ajustando automaticamente a capacidade de infraestrutura para atender à demanda.
- Alta Disponibilidade: Com Kubernetes, é possível distribuir os containers em diferentes zonas de disponibilidade, garantindo que o sistema continue funcionando mesmo em caso de falha de uma zona.
- Automação: Kubernetes automatiza processos como balanceamento de carga, failover e provisionamento, reduzindo o esforço operacional.

Consequências:

- A infraestrutura da plataforma se torna mais resiliente e preparada para crescer conforme necessário, com a capacidade de se adaptar dinamicamente às flutuações de uso.

9.4 Decisão 4: Uso de Azure SQL Server para Gerenciamento de Dados Relacionais

Decisão: O Azure SQL Server foi escolhido como o banco de dados relacional principal para o armazenamento de dados críticos, como perfis de usuários e transações financeiras.

Motivação:

- Escalabilidade Horizontal: O Azure SQL Server oferece uma arquitetura distribuída que permite escalar horizontalmente, suportando o crescimento da plataforma sem

comprometer a performance.

- Alta Disponibilidade: A replicação automática dos dados entre diferentes regiões e a capacidade de lidar com falhas garantem a alta disponibilidade, essencial para o funcionamento contínuo da plataforma.
- Consistência de Dados: A plataforma precisa garantir a integridade e a consistência das informações de transações financeiras e agendamentos, o que o Azure SQL Server fornece de maneira eficaz.

Consequências:

- A plataforma pode lidar com grandes volumes de dados e suportar um número crescente de usuários sem comprometer a performance ou a consistência dos dados.

9.5 Decisão 5: Utilização de MongoDB para Dados Não Estruturados

Decisão: O MongoDB foi escolhido para armazenar dados não estruturados, como preferências e avaliações dos usuários.

Motivação:

- Flexibilidade de Schema: O MongoDB permite que os dados sejam armazenados sem a rigidez de um schema pré-definido, facilitando o armazenamento de informações diversas e mutáveis, como feedback de turistas e guias.
- Alta Escalabilidade: MongoDB é otimizado para escalabilidade horizontal, permitindo que a plataforma gerencie grandes volumes de dados não estruturados de maneira eficiente.
- Geolocalização: MongoDB tem suporte nativo para operações geoespaciais, essenciais para a funcionalidade de mapas e localização de guias e turistas.

Consequências:

- A plataforma será capaz de lidar com dados dinâmicos e em constante mudança de maneira eficiente, mantendo a flexibilidade necessária para atender a diferentes tipos de dados dos usuários.

9.6 Decisão 6: Integração com APIs Externas para Mapas e Pagamentos

Decisão: A plataforma será integrada com o Google Maps para exibição de mapas e rotas, e com o Mercado Pago para processamento de pagamentos.

Motivação:

- Confiabilidade: Ambos os serviços são amplamente utilizados e oferecem alta confiabilidade, garantindo que os mapas e as transações financeiras funcionem de maneira estável.
- Segurança: O Mercado Pago fornece um sistema de pagamento seguro, em conformidade com as regulamentações de proteção de dados, enquanto o Google Maps oferece dados geoespaciais precisos e atualizados.
- Facilidade de Integração: As APIs fornecidas por ambos os serviços são bem documentadas e oferecem suporte a integrações eficientes com o sistema.

Consequências:

- A plataforma se beneficiará da confiabilidade e segurança oferecida por esses serviços externos, sem a necessidade de implementar internamente esses recursos complexos.

9.7 Decisão 7: Uso de Keptn para Automação de CI/CD

Decisão: O Keptn foi escolhido para gerenciar pipelines de CI/CD (Continuous Integration/Continuous Deployment) da plataforma.

Motivação:

- Automação: O Keptn automatiza a entrega contínua de novas funcionalidades e atualizações, garantindo que as implantações sejam feitas de forma segura e eficiente.
- Monitoramento e Testes Automatizados: O Keptn permite que testes automatizados sejam executados em cada etapa do pipeline, garantindo a qualidade das versões entregues.
- Rápida Iteração: A automação de CI/CD permite que a equipe de desenvolvimento faça atualizações rápidas e contínuas sem comprometer a estabilidade do sistema.

10. Requisitos de Qualidade

Os requisitos de qualidade são essenciais para assegurar que a plataforma “Take-a-Guide” atenda aos seus objetivos estratégicos e proporcione uma experiência confiável e eficiente para todos os usuários. Estes requisitos se baseiam no padrão ISO/IEC 25010, a seguir são descritos os principais requisitos de qualidade, juntamente com seus cenários de avaliação.

10.1 Segurança

A segurança é um requisito fundamental para proteger os dados dos usuários e garantir integridade das transações financeiras. Isso inclui a segurança de dados pessoais, a conformidade com a LGPD, proteção contra ataques e segurança das comunicações.

Cenário: Autenticação segura

- **Atributos de Qualidade (ISO 25010:2023)**: Segurança, conformidade

- **Decisão Arquitetural**:
    - Opção 1: Autenticação via Oauth 2.0 com suporte a autenticação multifator (MFA)
    - Opção 2: Autenticação por sistema proprietário com criptografia personalizada e autenticação de dois fatores por SMS ou e-mail
- **Trade-offs**:
    - Oauth 2.0 é um padrão amplamente utilizado, o que facilita a integração com outras plataformas, mas pode exigir configurações adicionais de segurança e licença de uso de plataformas externas.
    - A solução proprietária oferece maior controle sobre o sistema, mas pode implicar em maiores custos de desenvolvimento e manutenção, além de possíveis problemas de escalabilidade e suporte a longo prazo.
- **Resultados**:
    - Oauth 2.0 garante maior conformidade com padrões de mercado, maior segurança com MFA, mas com complexidade de implementação e possível dependência de terceiros.
    - Sistema proprietário pode fornecer flexibilidade e controle, porém com maiores custos iniciais e risco de vulnerabilidades se não for bem projetado.

Cenário: Criptografia de dados

- **Atributos de Qualidade (ISO 25010:2023)**: Segurança, conformidade
- **Decisão Arquitetural**:
    - Opção 1: Criptografia AES-256 para dados em repouso e TLS 1.3 para dados em trânsito
    - Opção 2: Criptografia RSA para dados em repouso e TLS 1.2 para dados em trânsito
- **Trade-offs**:
    - AES-256 é um padrão robusto, amplamente aceito e eficiente para criptografia em repouso, enquanto o TLS 1.3 é mais eficiente e seguro em comparação com o TLS 1.2.
    - RSA oferece alta segurança, mas é menos eficiente e pode gerar sobrecarga em sistemas com muitos dados. TLS 1.2 ainda é amplamente utilizado, mas menos seguro que a versão 1.3.
- **Resultados**:
    - AES-256 com TLS 1.3 oferece melhor desempenho e segurança, mas pode exigir upgrades em infraestrutura de sistemas antigos.
    - RSA com TLS 1.2 pode aumentar a latência e o custo computacional, além de não estar atualizado com os padrões mais seguros para transmissão de dados.

Cenário: Conformidade com LGPD

- **Atributos de Qualidade (ISO 25010:2023)**: Segurança, conformidade, privacidade
- **Decisão Arquitetural**:
    - Opção 1: Implementação de ferramentas de gerenciamento de consentimento (Consent Management Platforms - CMPs) integradas ao sistema
    - Opção 2: Implementação manual de coleta de consentimento e controle de dados com armazenamento em banco de dados próprio
- **Trade-offs**:

    - As CMPs automatizam a conformidade com a LGPD e simplificam a gestão de consentimento, mas têm custo adicional e podem depender de fornecedores externos.
    - A solução manual é flexível e personalizável, mas aumenta o custo de manutenção e pode ser suscetível a erros humanos, comprometendo a conformidade.
- **Resultados**:
    - CMPs garantem conformidade eficiente e automatizada, mas com dependência de terceiros.
    - Implementação manual oferece controle total, mas com risco elevado de não conformidade e maior carga de trabalho para desenvolvimento.

10.2 Escalabilidade

A plataforma deve ser capaz de suportar um número crescente de usuários, tanto turistas quanto guias, sem perda de desempenho. A escalabilidade horizontal deve ser assegurada pela infraestrutura baseada em containers e gerenciamento de recursos.

Cenário: Escalabilidade Automática

- **Atributos de Qualidade (ISO 25010:2023)**: Escalabilidade, desempenho
- **Decisão Arquitetural**:
    - Opção 1: Uso de Kubernetes para escalabilidade horizontal automática
    - Opção 2: Uso de serviços de escalabilidade automática integrados da AWS (Elastic Beanstalk) ou Google Cloud (App Engine)
- **Trade-offs**:
    - Kubernetes oferece flexibilidade e controle sobre o escalonamento, mas exige maior expertise e configuração complexa.
    - Serviços de escalabilidade integrados (AWS/GCP) simplificam a escalabilidade, mas limitam o controle sobre a infraestrutura e podem ser mais caros no longo prazo.
- **Resultados**:
    - Kubernetes permite um controle refinado e customização, mas com maior complexidade e curva de aprendizado.
    - Serviços de nuvem simplificam a configuração e reduzem a carga de manutenção, porém com menor flexibilidade e custo potencialmente mais alto.

10.3 Disponibilidade

Descrição:

O sistema "Take-A-Guide" deve estar disponível 24/7 para garantir que turistas e guias possam acessar a plataforma em qualquer momento. A alta disponibilidade é garantida por meio da replicação de dados e da distribuição de componentes em diferentes zonas de disponibilidade.

Cenários:

Resiliência a Falhas: O sistema deve ser capaz de continuar operando mesmo em caso de falha de um ou mais servidores, garantindo que os usuários não experimentem interrupções no serviço.

Monitoramento Contínuo: Ferramentas como Prometheus e Grafana devem monitorar a

plataforma continuamente, detectando falhas e ativando automaticamente instâncias de backup em caso de problemas.

10.4 Desempenho

Descrição:
A plataforma deve garantir tempos de resposta rápidos, mesmo em picos de uso. Isso é fundamental para manter uma experiência de usuário fluida e evitar frustrações, principalmente durante operações críticas, como pagamentos ou reservas.
Cenários:
Tempo de Resposta Rápido: O tempo de resposta para a maioria das operações (como login, busca de guias, e carregamento de perfis) deve ser inferior a 2 segundos.
Otimização de Performance: O sistema deve utilizar técnicas de otimização de cache e consultas ao banco de dados para garantir uma performance consistente, mesmo com aumento de carga.

10.5 Portabilidade

Descrição:
A plataforma deve ser acessível em diversos dispositivos e navegadores, garantindo uma experiência consistente independentemente do ambiente de acesso.
Cenários:
Suporte Multi-Dispositivo: A plataforma deve ser testada e compatível com os principais navegadores (Chrome, Firefox, Safari) e dispositivos (desktop, tablet, mobile).
Adaptabilidade: A interface deve ajustar-se automaticamente ao tamanho da tela do dispositivo, sem perda de funcionalidade ou usabilidade.

11. Riscos Técnicos

Um dos principais riscos é a compatibilidade de versões. Com a constante evolução de bibliotecas e frameworks, é comum que novas versões introduzam mudanças que podem quebrar funcionalidades existentes. Para mitigar esse risco, as equipes devem realizar testes rigorosos após cada atualização e documentar as versões utilizadas. Essa prática ajuda a garantir que a aplicação permaneça estável e funcional. Além disso, a dependência de bibliotecas externas representa um risco significativo, se uma biblioteca for descontinuada ou não receber atualizações de segurança, isso pode expor a aplicação a vulnerabilidades. Os dados armazenados na nuvem podem ser vulneráveis a ataques cibernéticos, como injeções, ransomware e violações de dados. É essencial implementar medidas de segurança robustas, como criptografia e autenticação multifator. A disponibilidade do serviço em nuvem pode ser afetada por falhas de infraestrutura ou problemas técnicos do provedor. Isso pode resultar em interrupções de serviço. Além disso, a integração de sistemas pode apresentar desafios, especialmente ao lidar com APIs externas. A qualidade dos dados é uma preocupação constante, pois dados inconsistentes ou incompletos podem comprometer a precisão das análises e relatórios. A falta de um plano de backup adequado pode resultar em perda de dados em caso de falhas, enquanto a dependência de fornecedor pode dificultar a migração para outras plataformas no futuro. É essencial uma documentação clara e atualizada, pois a falta dela pode complicar a manutenção e a evolução do sistema.