Documento Arquitetural

API Controle de Estoque de Bebidas

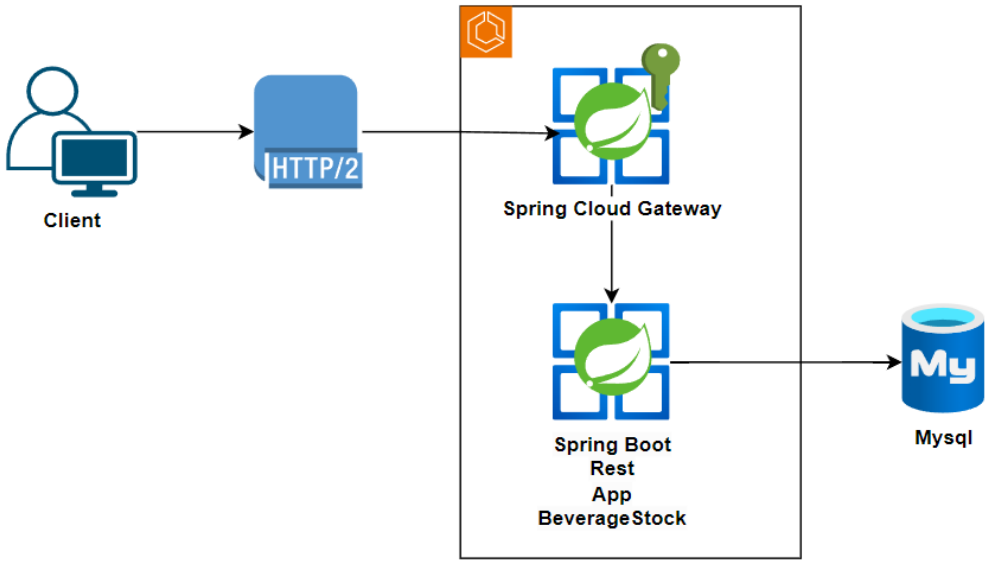
Contexto

Uma empresa do ramo de marketplace possui um depósito de bebidas com seções distintas para armazenamento de bebidas, onde se faz necessário a criação de uma API para gestão de dados deste depósito. Atualmente o estoque é responsável por armazenar dois tipos de bebidas (alcoólicas e não alcoólicas) e o sistema deve garantir que as regras de negócio sejam aplicadas corretamente e que possa escalar conforme as necessidades futuras.

Objetivo

Este documento apresenta uma sugestão de arquitetura para a criação de uma API REST Full destinada a gerenciar os dados do armazenamento e estoque de um depósito de bebidas, a API deve permitir o cadastro e consulta de bebidas, controlar o estoque e fornecer histórico de entradas e saídas, tudo dentro de regras de negócio específicas para garantir a integridade dos dados e operações.

Visão Geral da Arquitetura



Especificação da Solução

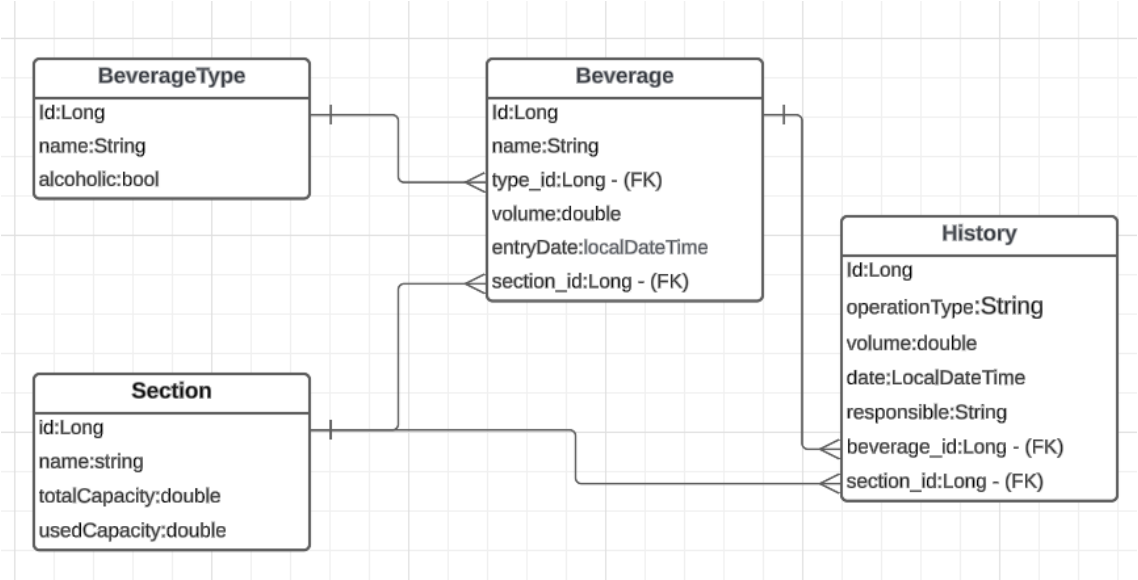
Requisitos Funcionais

ID	Descrição Resumida
RF01	O sistema deve permitir o cadastro de bebidas em uma seção específica do estoque.
RF02	O sistema deve permitir a consulta de bebidas armazenadas em cada seção.
RF03	O sistema deve permitir a consulta do volume total de bebidas armazenadas no depósito por tipo de bebida.
RF04	O sistema deve permitir a consulta das seções que possuem espaço suficiente para armazenar uma quantidade específica de bebida.
RF05	O sistema deve permitir a consulta das seções que possuem bebidas disponíveis para venda, baseadas no tipo de bebida.
RF06	O sistema deve permitir o registro de operações de entrada e saída de bebidas no histórico de operações.
RF07	O sistema deve permitir a consulta do histórico de operações de entrada e saída de bebidas, com possibilidade de ordenação por data e seção.
RF08	O sistema deve garantir que uma seção não possa armazenar mais de um tipo de bebida simultaneamente.
RF09	O sistema deve garantir que todas as operações de entrada e saída sejam registradas no histórico.
RF10	O sistema deve garantir que uma seção não possa receber bebidas não alcoólicas se armazenou alcoólicas no mesmo dia.

Requisitos Não Funcionais

ID	Descrição Resumida	Classificação
RNF01	A API deve ser otimizada para responder rapidamente a consultas, especialmente na consulta de volumes e seções disponíveis	Desempenho
RNF02	A arquitetura deve permitir a adição de novos tipos de bebidas sem grandes mudanças no sistema	Escalabilidade
RNF03	O código deve seguir os princípios da Clean Architecture e SOLID, facilitando futuras manutenções e evoluções do sistema	Manutenibilidade
RNF04	A API deve implementar autenticação e autorização para proteger as operações críticas, como o registro de entradas e saídas	Segurança
RNF05	Deve haver uma cobertura robusta de testes unitários e de integração para garantir a confiabilidade do sistema	Testabilidade

Modelagem de Entidades

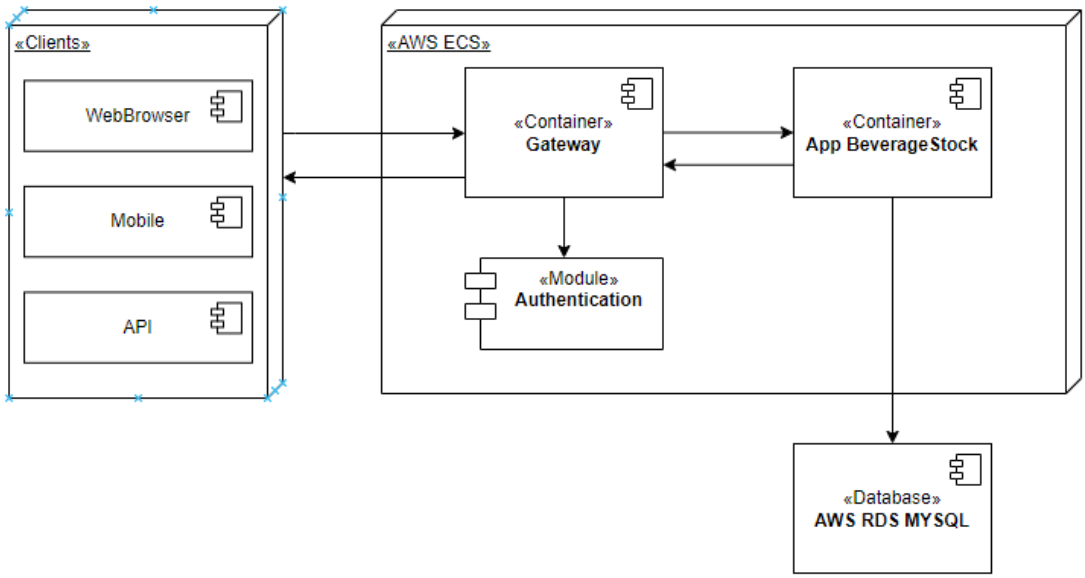


Beverage ↔ BeverageType: (Muitos para Um) Cada bebida está associada a um único tipo de bebida.

Section ↔ Beverage: (Um para Muitos) Cada seção pode conter muitas bebidas, mas cada bebida só pode estar em uma seção.

Section ↔ History: (Um para Muitos) Cada seção pode ter muitos registros de histórico, mas cada registro de histórico pertence a uma seção.

Infraestrutura



«Clients» Web Browser/Mobile/API: Representação dos possíveis clientes de API com possibilidade a diversas interfaces como navegador web, API de terceiro ou aplicativo mobile.

«AWS ECS»: Elastic Container Service é um serviço gerenciado de orquestração de contêineres oferecido pela AWS que permite execução, gerenciamento e dimensione aplicações containerizadas usando contêineres Docker na infraestrutura da AWS.

«Container Gateway»: Porta de entrada da aplicação onde deve haver autenticação e roteamento para o serviço solicitado.

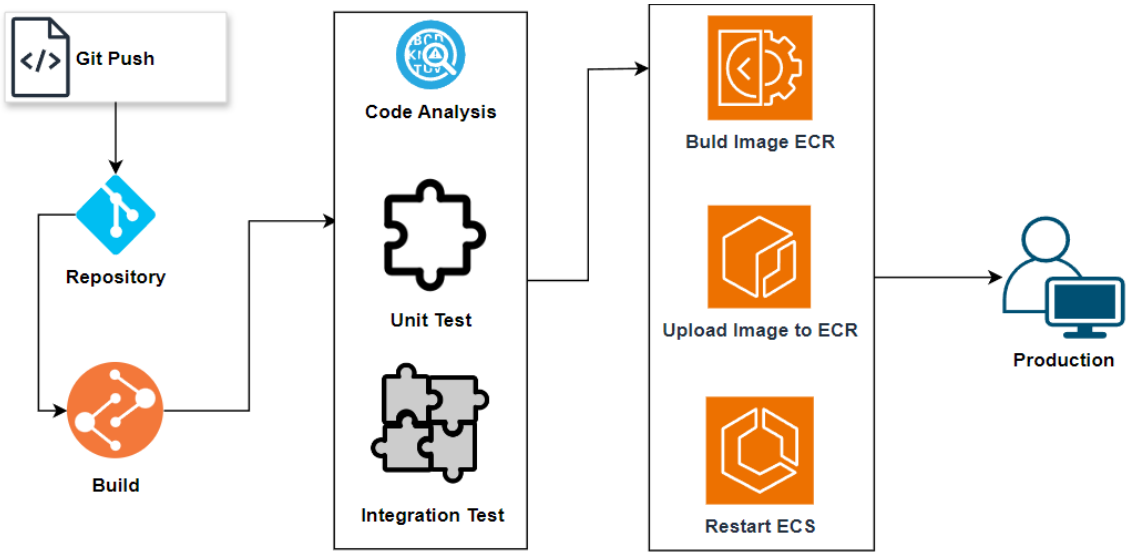
«Container Gateway»: Porta de entrada da aplicação onde deve haver autenticação e roteamento para o serviço solicitado.

«Module Authentication»: Modulo de autenticação acoplado ao gateway, que deve ser responsável por autorizar as requisições.

«Container AppBeverageStock»: Aplicação principal onde é realizada toda a gestão de dados do estoque de bebidas.

«Database AWS RDS MYSQL »: Relational Database Service MySQL é um serviço gerenciado de banco de dados relacional oferecido pela AWS que simplifica o processo de configuração, operação e escalabilidade de bancos de dados MySQL na nuvem.

Deploy, Testes e Qualidade



Pipeline de CI/CD: Implementar um pipeline de CI/CD utilizando GitHub Actions para automação de build, testes e deploy de imagens em serviços como AWS ECR(Registro de imagens de container) e deploy no AWS ECS.

Ambientes: Configurar ambientes separados para desenvolvimento, staging, e produção, com deploy automatizado para cada um.

Frameworks para Testes: Utilizar frameworks como JUnit, Spoock e Mockito para escrever testes unitários e integração.

Testes de Unidade: Implementar testes de unidade das entidades de domínio e casos de uso se utilizando de mocks para simular repostas a serviços externos.

Testes de Integração: Implementar testes de integração para garantir que os repositórios e controladores interajam corretamente com o banco de dados e quando necessário utilizar de TestContainers para criar instâncias temporárias do banco de dados.

Ferramentas de Qualidade: Utilizar SonarQube para medir a cobertura de testes no código e possíveis vulnerabilidades integrando o relatório ao pipeline de CI/CD.