Microservices (Hands-ON)

ANOTAÇÕES TÉCNICAS - ALURA

FIAP SHIFT

2023

Abril, 2023

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

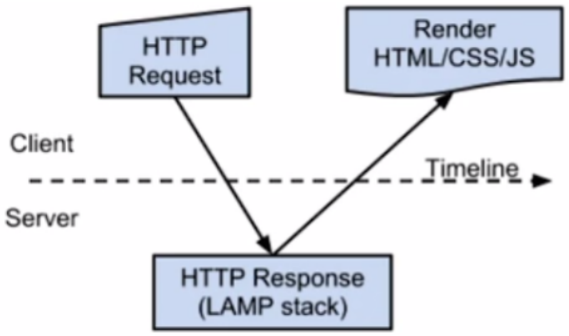
ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

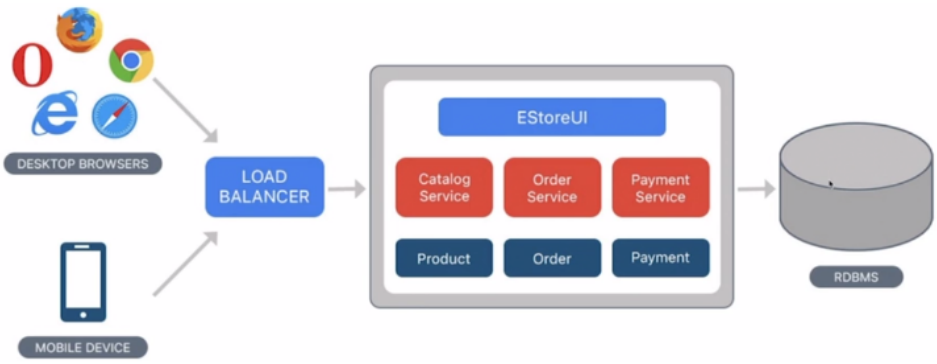
Conhecendo a arquitetura

Como funciona a web?

- A internet funciona através de um protocolo HTTP



- Aplicações monolíticas



Alguns problemas:

- Demoras no deploy, então um erro em uma parte da aplicação pode quebrar uma outra que não tem nenhuma relação.
- Falhas podem derrubam o sistema todo.
- 1 projeto = 1 tecnologia

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Arquitetura de microsserviços

Microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas. Esses serviços pertencem a pequenas equipes autossuficientes.

Algumas vantagens

- Projetos independentes = tecnologias independentes
- Falha em 1 serviço é isolada
- Deploys menores e mais rápidos

Algumas desvantagens

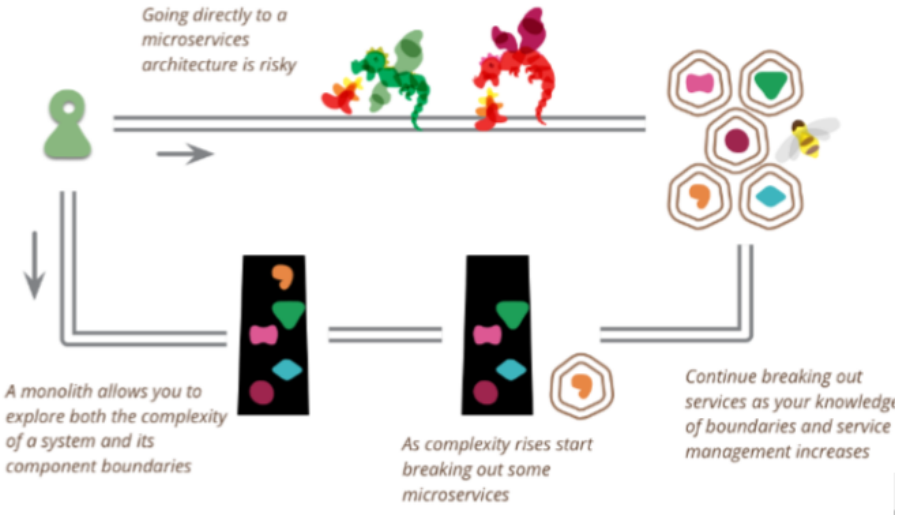
- Maior complexidade de desenvolvimento e infra
- Debug mais complexo
- Comunicação entre os serviços deve ser bem pensada
- Diversas tecnologias pode ser um problema
- Monitoramento é crucial e mais complexo

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Monólito por padrão



Tipos de microsserviços

- Data Service
- Business Service
- Translation Service
- Edge Service

Separando serviços

Serviços de Domínio: Fornece acesso a um único domínio da aplicação e lá suas regras estão contidas.

- Domain-driven Design
- Comece modelando seu domínio, não pensado na persistência
- Avalie as ações que serão disponibilizadas
- Construa o serviço, pensando primeiro o contrato
- REST e RPC podem andar juntos

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Serviços de negócio: Em determinados momentos as operações precisam de mais de um modelo do nosso domínio para serem corretamente representadas em um serviço

- Proveem uma funcionalidade do negócio de mais alto nível
- Permite encapsular domínios relacionados

Criando um serviço de negócio

- Identifique o processo que você pretende expor
- Identifique os domínios que serão necessários nesse serviço
- Defina a API que será utilizada, focando no domínio e não nos dados
- Consuma serviços de domínio para executar os processos

Estrangulando um monolito, ou Strangler pattern

- Quebrar um monolito, tirando as funcionalidades dele
- Podemos começar isolando os dados
- Ou podemos começar isolando o domínio

Sidercar pattern

- Determine o processo comum
- Construa um módulo compartilhável
- Aplique esse sidecar nos serviços que precisam dele

Integrando serviços

Ponto único de entrada

API Gateway

- Problema: Clientes acessando livremente os serviços geram caos
- Gateway fornece um proxy, uma fachada, para as necessidades reais
- Desvantagens: Esse portão de entrada pode se tornar um ponto central de falha

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Comportamentos do Gateway
- Simplesmente autorizar e redirecionar os requests
- Uso de Decorator para adicionar informações necessárias aos requests
- Limitar o acesso ou conteúdo trafegado

Agregando processos

Process aggregator pattern
- Serviços de negócio agregam serviços de domínio
- Process aggregators agregam serviços de negócio
- Agregadores fazem as chamadas para os serviços necessários e montam a resposta correta
- Pode (e deve) ter lógica de processamento

Construindo um agregador
- Defina um novo modelo para representar os dados agregados
- A partir desse modelo, pense na API que fornecerá as operações

Edge pattern
- Gateway específico para determinado(s) cliente(s)
- Foco nas necessidades reais de determinados clientes

Construindo uma ponta
- Identifique o cliente e suas necessidades
- Construa contratos específicos para o cliente
- Modifique os dados que são transferidos para garantir a otimização do processo
- Existe a possibilidade de ter apenas Edges, e não Gateways

Lidando com dados

Single service database
- Problema: Escalabilidade do serviço e do banco são fortemente relacionados
- Solução: Cada serviço (que precisar) terá seu próprio banco de dados

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Shared service database

- Problema: Às vezes precisamos centralizar os dados (até por motivos contratuais)
- Solução: Trate esse banco em cada serviço como se ele fosse separado

Padrões de codificação

CQRS (Command Query Responsibility Segregation), trata-se de um padrão arquitetural escalável, propondo a separação das responsabilidades em canais de comunicação distintos, descritos como modelo de Escrita (command) e leitura (query).

- Com leitura e escrita separados, cada parte pode realizar operações mais complexas
- O modelo de leitura pode ter informações agregadas de outros domínios
- O modelo de escrita pode ter dados sendo automaticamente gerados
- Aumenta (MUITO) a complexidade de um sistema

Eventos assíncronos

Asynchronous eventing

- Determinados problemas NÃO PODEM ser resolvidos na hora (em tempo real)
- Um serviço emite um evento que será tratado em seu devido tempo
- Tecnologias como mensagerias e serviços se stream de dados brilham

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: PADRÕES DE PROJETO

Operações

Agregando logs

- Formatos de log DEVEM ser compartilhados entre os serviços
- Uma taxonomia comum deve ser compartilhada
- Logs de monolitos são agregados por padrão. Com microsserviços o buraco é mais embaixo
- Parte da tarefa de agregação pode ser o parsing dos logs para categorizar corretamente

Rastreando chamadas

- Uma parte importante de realizar logs é rastrear as chamadas de uma execução
- Devemos poder reconstruir uma operação a partir de um identificador
- Isso é o equivalente à call stack de um sistema monolítico
- Use padrões de trace ID para gerar os logs
- Use ferramentas de gerenciamento (APMs) para visualizar

Agregando métricas

- Enquanto logs precisam de desenvolvimento, métricas “só” precisam de instrumentação
- Métricas nos permitem saber o que está acontecendo em determinado momento
- Construa ou use dashboards de alto nível para ter uma fácil visão do status atual da aplicação
- Depois, tenha dashboards específicos para cada serviço, com mais detalhes

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Arquitetura microsserviços

Benefícios dos microsserviços:
- Microsserviços e a agilidade
- Escalabilidade flexível
- Fácil implantação
- Liberdade tecnológica
- Microsserviços e o código reutilizável
- Resiliência

O que é uma arquitetura monolítica?
- É uma modelagem de desenvolvimento de software que cria serviços dependentes uns dos outros, em uma única estrutura. Qualquer falha pode comprometer toda a estrutura.

De que é composto um microsserviço?
- Sabemos que cada microsserviço deve ser o dono e gerenciar seus próprios dados.

Então será que um microsserviço é um único processo rodando em um único servidor?

A nossa aplicação principal, precisa se comunicar com um servidor de banco de dados

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

O microsserviço pode realizar tarefas além do que a sua interface pública permite e além do que serviços externos podem solicitar para ele fazer.

- Uma máquina (servidor) pode ser considerada um componente. Várias aplicações em uma mesma máquina podem ser vários componentes. Um serviço de apoio (como banco de dados ou fila de mensageria) pode ser um componente. Qualquer coisa que efetivamente componha o serviço, é um componente.

Microsserviços são independentes:

- Um microsserviço expõe alguma forma de comunicação (uma API). Isso é um contrato entre este microsserviço e seus clientes.

Mas como manter a minha API sempre a mesma se meu projeto precisa de atualizações e novas funcionalidades?

- Apenas faça modificações aditivas
  - Novos endpoints
  - Novos campos (opcionais) em cada recurso
- Versionamento de APIs
  - Ao lançar uma v2, a v1 deve continuar funcionando, inalterada
- Manter equipes separadas, donas de cada serviço
  - A mesma equipe vai alterar os clientes
  - Para adicionar funcionalidades que dependam de outros, solicitações formais podem ser feitas

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

API (Application Programming Interface):

- Se uma aplicação consegue se expor a outras, através de HTTP, por exemplo, provendo acesso a funcionalidades, podemos chamá-la de API. Essa aplicação fornece uma interface de programação para outras.
- Um método ou um conjunto de métodos públicos fazem parte da API de uma classe. Isso é o que chamamos de interface pública, ou seja, o que está acessível a partir de outras classes. É um termo pouco usado, porém correto.

Antes de implementar:

- Desenhe um fluxo real usando uma arquitetura de microsserviços. Desta forma os problemas de cada abordagem surgirão

Arquitetura:

- MVC (Model-View-Controller)
- ADR (Action-Domain-Response)
- SOA (Arquitetura orientada a serviços)
- Clean Architecture
- Hexagonal Architecture
- Onion Architecture

Criação de serviços

Cuidando do host, como iremos lidar com os servidores?

- Onde manter cada componente de um microsserviço?
  - Já vimos que cada microsserviço pode ter mais de um componente.
- Onde podemos manter todo esse serviço de forma que facilite tanto o desenvolvimento quanto o deploy?

Máquinas virtuais:

- O uso de máquinas virtuais foi e ainda é muito comum. Podemos provisionar uma VM de forma automatizada e integrar várias VMs.
- O custo de processamento é alto. Dificilmente um computador de desenvolvimento rodará tranquilamente dezenas de VMs.

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Sistemas em cloud:

- Sistemas de computação em nuvem estão cada dia mais famosos. Manter um ambiente de produção em cloud é relativamente simples hoje em dia.
- Porém como ter um ambiente de desenvolvimento simplificado? Precisamos contratar uma máquina para cada dev?

Containers:

- Essa é hoje a opção mais recomendada. Um ambiente de desenvolvimento é facilmente criado usando containers e diversos serviços de cloud possuem “container hosts”, ou seja, ambientes próprios para “subirmos” containers.
- Uma ótima forma de ter uma espécie de “template” e tendo uma imagem base para containers que conterão microsserviços
- A partir da imagem, só precisamos começar a codificar e rodar o código nos containers criados

Qual a principal diferença entre o uso de Containers e Máquinas Virtuais?

- Consumo de recursos.

- Containers conseguem compartilhar recursos com o sistema operacional host, enquanto cada máquina virtual é um novo sistema operacional. Isso exige muito menos recursos.

Criando um novo serviço: Quais as etapas necessárias para criar um novo serviço?

- Antes de qualquer coisa, precisamos configurar o repositório para versionarmos nosso código.
- Aqui devemos nos perguntar se vamos optar por uma abordagem monorepo, por exemplo
- Integração e entrega contínuas (CI e CD) são praticamente obrigatórias ao se trabalhar com microsserviçõs.
- Devemos ter todo o processo automatizado e testes devem não só existir, mas devem ser confiáveis.
- Um padrão sempre deve ser seguido para que todas as equipes estejam aptas a criar novos serviços.

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Definindo um padrão

Diversas tarefas devem ser realizadas de forma semelhante entre os serviços:

- Criação de logs
  - Formato
  - Destino
- Verificações de status (health checks)
- Monitoramento de métricas
- Busca por configuração e secrets

Templates ou exemplos

- Podemos ter um projeto “esqueleto” com tudo que qualquer microsserviço precisa.
- Scripts de build, mínimo código necessário, etc.
- Isso nos dá muita agilidade, mas tira um pouco da flexibilidade

Qual a desvantagem de seguir padrões ao criar novos serviços?

- Menos flexibilidade.
  - Ao definirmos padrões (como de linguagem ou ferramentas), abrimos mão de certa flexibilidade que a arquitetura de microsserviços nos traz. Isso deve ser posto "na balança".

Como se comunicar

Como lidar com a comunicação entre serviços?

- Todo serviço pode se comunicar com qualquer outro?
- O front-end deve chamar os serviços diretamente?
- Regras: não há regras

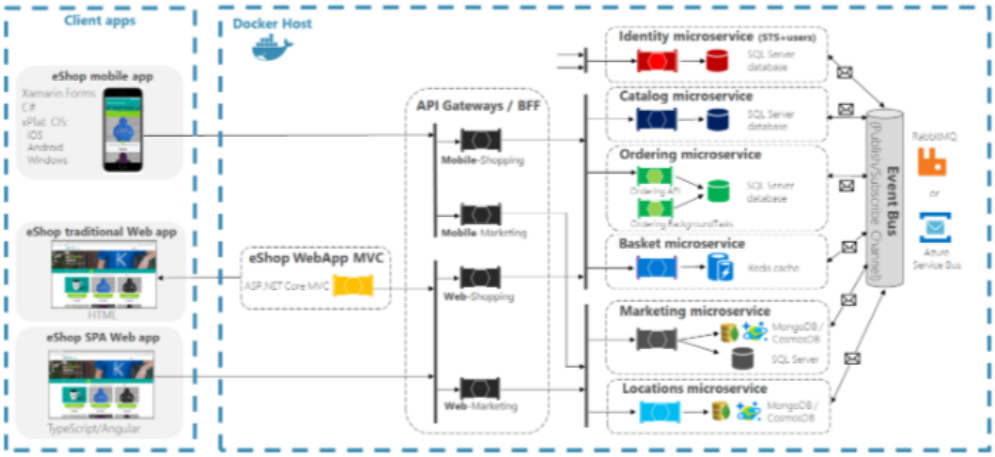
Possíveis problemas

- Dependências descontroladas
- Falhas em cascata
- Performance prejudicada

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS



O API Gateway, ou porta de entrada, tem como função facilitar o roteamento, monitoramento e política de acessos aos microsserviços.

Uma vantagem do uso de um API Gateway?

- Podemos monitorar toda a aplicação de um único ponto.
  - Através do API Gateway podemos monitorar acessos a nossa aplicação, podemos ter uma ideia geral de erros que estejam acontecendo, monitorar performance, etc.

Comunicação síncrona:

- Comunicação direta: Diversos cenários nos obrigam a realizar “chamadas” e esperar por suas respostas
- Isso é o que conhecemos como comunicação síncrona
- Como se comunicar
  - HTTP
  - gRPC
  - Protocolos personalizados
- Problema da abordagem: Ao realizar uma chamada direta para outro serviço esperar sua resposta, problemas neste outro serviço nos afetarão diretamente.

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Comunicação assíncrona:

- Comunicação indireta: Existem cenários onde a resposta não precisa ser obtida imediatamente
- Como se comunicar
  - CQRS (background tasks)
  - Eventos (mensageria)
- Exemplo simples: Ao realizar uma compra
  - Dados precisam ser validados
  - Pagamento deve ser processado
  - Estoque deve ser separado
  - Logística deve ser iniciada

Quando usar comunicação síncrona e quando usar assíncrona, respectivamente?

- Quando precisamos da resposta imediatamente e quando podemos "ignorar" a resposta no momento.
  - Se precisamos obter a resposta na hora antes de continuar o processamento, precisamos usar comunicação síncrona. Para exibir os dados de um curso, por exemplo, precisamos nos comunicar de forma síncrona com o banco para obter os dados e exibi-los. Já para enviar um e-mail, não precisamos ficar esperando a confirmação. Podemos enviar o pedido e ser notificados depois se deu certo ou não. Nestes cenários podemos usar comunicação assíncrona.

Lidando com falhas

- Como diz a lei de Murphy...
- Microsserviços possuem operações internas em rede. A possibilidades de falha são grandes
- Falhas em comunicação síncrona
  - Circuit breaker
  - Cache
- Falhas em comunicação assíncrona
  - Simples Retry
  - Retry com back-off
  - Fila de mensagens mortas
  - Mensagens devem poder ser lidas fora de ordem
  - Mensagens devem poder ser recebidas repetidamente (idempotência)

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Service Discovery

- Microsserviços podem estar na mesma rede ou em redes separadas, e cada serviço pode estar exposto por um IP
- Mas lidar diretamente com o IP de cada serviço pode trazer muitos problemas
- DNS

- Algo que já fazemos na internet, podemos fazer também em redes privadas: Registro de nomes

- DNS, pode ser utilizado como service registry para sabermos como acessar cada serviço

- Um registro de serviço pode ter informações sobre quais processos ou máquinas estão de pé e sem falha

Segurança de serviços

Segurança geral

- Segurança no transporte
  - HTTPS
- Segurança no repouso
  - Criptografia
    - Criptografia de disco
    - Bancos de dados cifrados
    - Criptografia em back-ups
  - Anonimização

TLS

- Transport Layer Security e é usada pelo protocolo HTTPS para garantir a segurança das mensagens.

Argon2

- é uma função de hash de senhas ganhadora de diversos prêmios. É uma das mais recomendadas para armazenamento de senhas cifradas.

Md5

- é um algoritmo de hash muito utilizado para verificar integridade de arquivos transferidos na rede, por exemplo, ou após serem transferidos entre dispositivos para backup.

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

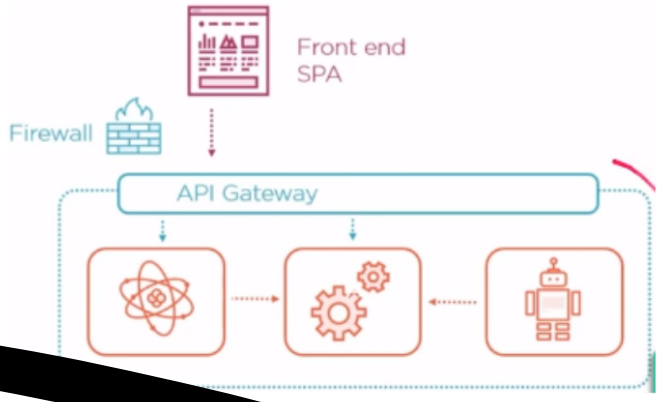
Autenticação e autorização

- Não devemos confiar em ninguém: Ter uma certa segurança (amplamente falado) não isenta nossa aplicação de lidar diretamente com esse assunto.
- Autenticação: Cada requisição deve informar quem é o cliente. A partir dessa informação, nossa aplicação pode decidir se a operação será realizada ou não.
- Técnicas de autenticação
  - Basic HTTP
  - Tokens (JWT)
  - OAuth
  - OpenID Connect
- A **autenticação** nos permite saber quem está realizando determinada chamada. A partir do processo de **autorização** decidiremos se a pessoa autenticada pode realizar tal ação
- Técnicas de autorização
  - ACL (Access control list)
  - RBAC (Role-based access control)
  - On behalf of

Segurança na rede

- Nem só de programação vive uma aplicação
- É de extrema importância que façamos uso de recursos de rede para manter nossa aplicação ainda mais segura, além do que já foi citado
- Rede virtuais

- Sistemas de firewall

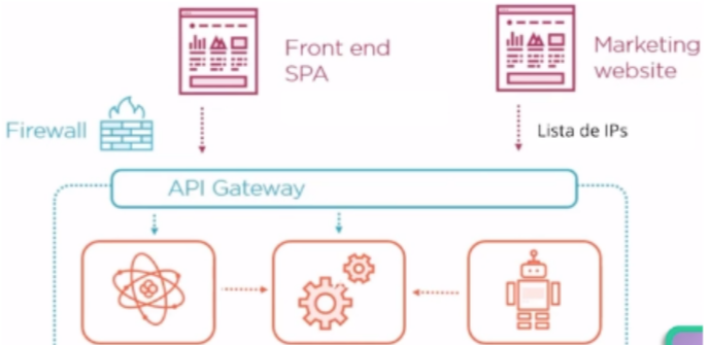


MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

- Lista de IPs permitidos



Anonimização

- faz parte da segurança em repouso, ou no destino.

Defense in depth

- Devemos dificultar ataques ao máximo: Todas as formas de segurança que foram citadas são muito importantes, mesmo que às vezes redundantes
- Como me proteger mais?

- Atacantes (hackers) utilizam ferramentas modernas. Conheça essas ferramentas! Estude os possíveis ataques que podem afetar seu sistema
- Tenha uma equipe de infosec e execute pentests
- Automatize verificações de segurança. Faça requisições com certificados inválidos, usuários não autorizados, etc
- Monitore e detecte ataques em tempo real
- Tenha logs e audite os sistemas com frequência

Lidando com o deploy

Entregas automatizadas

- Um monolito pode ser entregue de forma manual, dependendo do caso. Um processo simples sendo executado com periodicidade pode atender alguns cenários
- Mas quando tratamos de microsserviços, isso se torna inviável
- Release pipeline:

- Uma release pipeline é uma linha de processos executados para gerar a entrega de um projeto
- Nesta pipeline podem estar processos de build, verificações, testes

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Por que um processo manual pode ser aceito em uma aplicação monolítica?

- Por ser uma única aplicação, com um único "roteiro" de deploy.
  - Quando falamos de microsserviços, cada serviço pode ter um requisito especial a ser cumprido na hora do deploy. É humanamente impossível realizar esta tarefa de forma manual. Por isso DEVEMOS automatizar o processo.

Ambientes de execução

- O microsserviço pode ser implantado em vários ambientes
  - Desenvolvimento
  - Staging/QA
    - Testes de performance
    - Smoke tests
  - Homologação
  - Produção
    - Por cliente
    - Por região
- Configurações parametrizadas
  - Configurações do ambiente em si
    - Quantidade de recursos
    - Localização
  - Configurações da aplicação
    - Destino de log
    - Dependências
    - Dados de acesso

Por que precisamos ter configurações parametrizadas em cada ambiente?

- Para podermos ter acesso a recursos específicos de cada ambiente sem tocar no código.
  - Imagina que para nos conectarmos ao banco de dados tenhamos os dados de acesso direto no código. Vamos precisar mudar este código na hora do deploy e desfazer a alteração na hora do desenvolvimento. Isso não é nada prático. Serviços também podem ser diferentes em ambientes diferentes. Podemos usar "versões" menos robustas em ambientes de desenvolvimento, por exemplo.

MICROSERVICES (HANDS-ON)
API GATEWAY, WEBHOOKS & ELASTIC APM

ANOTAÇÕES TÉCNICAS - ALURA

MICROSSERVIÇOS: EXPLORANDO OS CONCEITOS

Estratégias de deploy

- Existem diversas estratégias para transformarmos uma entrega em um deploy, e várias etapas para cada estratégia
- Do que fazer deploy?
  - Arquivo distribuivel (.exe, .war, .phar, .zip)
  - Tag do git
  - Imagem do container

- Serviços são independentes: É importante que nossa pipeline possa realizar a entrega e deploy de microsserviços de forma isolada e não precisemos fazer o build de todos os serviços da aplicação
- Estratégias de releases
  - Rolling upgrade
  - Blue-green
  - Feature toggle