Treinamento Cloud Fundamentos Tecnológicos

Criador: Neuber Paiva de Souza

Data de criação: 10/10/2023

Versão: v1.1

Sumário

Fundamentos Tecnológicos

1. Conceitos das stacks Front-End, Back-End e Full Stack

Os projetos de softwares, principalmente os voltados para a web, são divididos em stacks, front-end e back-end. Onde basicamente, o front-end é tudo aquilo que o usuário de um site, sistema ou aplicativo pode ver e o back-end é a engrenagem que faz tudo acontecer nos bastidores. Diferentes tecnologias se aplicam a cada uma destas stacks.

Já, o full-stack, como o próprio nome sugere, é a união das habilidades tanto em back-end como em front-end, além do conhecimento com integrações e bancos de dados, os desenvolvedores full-stacks são uma peça importante no time de desenvolvimento.

1.1. Fundamentos e tecnologias utilizadas no Front-End

O desenvolvedor front-end é responsável pela parte visual, criação e disponibilização dos artefatos relacionados aos conteúdos, funcionalidades e informações que são exibidas aos usuários da aplicação.

Neste treinamento, vamos aprender as seguintes linguagens, frameworks e tecnologias:
HTML, Javascript, CSS, TypeScript, Angular e React.

1.2. Fundamentos e tecnologias utilizadas no Back-End

O desenvolvedor back-end é o responsável pelo funcionamento da aplicação atrás das telas, deixando o seu conteúdo dinâmico e estruturado com o apoio de um banco de dados e serviços externos (integrações).

O profissional de back-end, precisa dominar linguagens, como o Java e o NodeJS, além de possuir conhecimentos avançados em bancos de dados e integrações com o REST.

1.3. Aprendendo os conceitos do Full Stack

O desenvolvedor full-stack, acomula os conhecimentos das 2 stacks anteriores, dominando tanto o front-end como o back-end, além de possuir conhecimentos avançados em bancos de dados, integrações, devops, etc.

Este profissional é conhecido como “coringa ou faz tudo”, pois ele tem a capacidade de resolver problemas e buscar as melhores soluções para os desafios técnicos, enfrentados pela equipe de desenvolvimento.

Para se tornar um profissional full-stack completo, além de desenvolver os conhecimentos técnicos chamados de hard skills, é necessários desenvolver algumas habilidades comportamentais as soft skills, podemos destacar as principais soft skills desejadas neste profissional:
Inteligência Emocional, Trabalho em Equipe, Colaboração, Concentração, Criatividade, Proatividade, Resiliência, Versatilidade, Constante Aprendizado, etc.

2. Aprendendo sobre Arquitetura

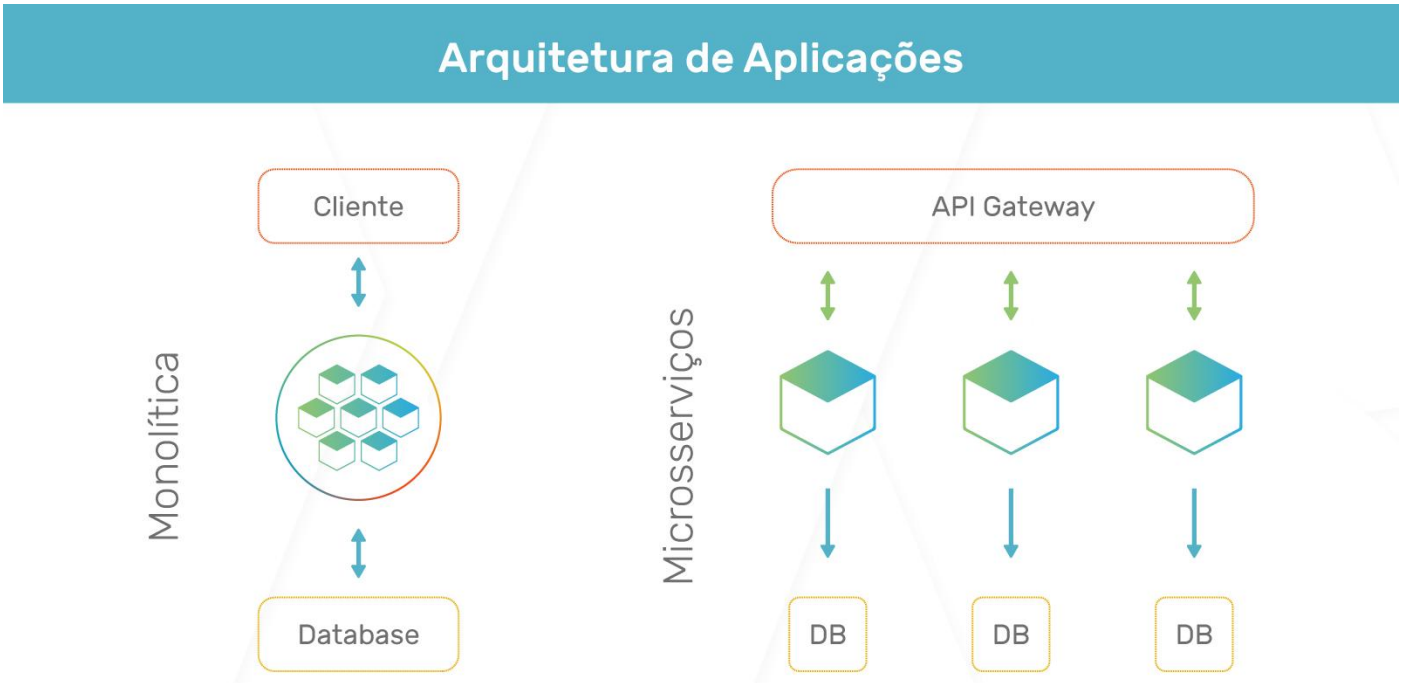
Existem diversos tipos de arquitetura na área de TI, algumas são mais direcionadas para a área técnica e estratégica de TI, outras têm um foco mais direcionado ao negócio. Vamos conhecer os principais tipos de arquitetura e que utilizaremos em nosso treinamento:

2.1. Arquitetura Monolítica

A Arquitetura Monolítica é um modelo tradicional de desenvolvimento de software que usa uma base de código para executar várias funções comerciais. Todos os componentes de software em um sistema monolítico são interdependentes devido aos mecanismos de troca de dados dentro do sistema. É restritivo e demorado modificar a arquitetura monolítica, pois pequenas mudanças afetam grandes áreas da base de código.

2.2. Arquitetura de Microsserviços

Na Arquitetura de Microsserviços utiliza-se uma abordagem arquitetônica que compõe o software em pequenos componentes ou serviços independentes. Cada serviço executa uma única função e se comunica com outros serviços por meio de uma interface bem definida. Como eles são executados de forma independente, você pode atualizar, modificar, implantar ou escalar cada serviço conforme necessário. Neste treinamento, utilizaremos o framework Spring Boot na criação dos nossos microsserviços.



2.3. Arquitetura Cloud

A Arquitetura Cloud (de Nuvem) é como todos os componentes e recursos necessários no desenvolvimento de uma nuvem são conectados para construir uma plataforma online em que as aplicações serão executadas. É como a construção de uma casa: a infraestrutura da nuvem inclui todos os materiais, enquanto a arquitetura da nuvem é a planta.

2.4. **Arquitetura Orientada a Eventos**

A Arquitetura Orientada a Eventos é um modelo de arquitetura para o design de aplicações em um sistema orientado a eventos, onde os componentes de captura, comunicação, processamento e persistência de eventos formam a estrutura básica da solução. Diferente do modelo tradicional orientado a solicitações, ela é composta de produtores e consumidores de eventos. Um produtor detecta ou percebe um evento e o representa como uma mensagem. Ele não conhece o consumidor nem o resultado do evento. Após um evento ser detectado, ele é transmitido do produtor para o consumidor por meio de canais, em que uma plataforma processa eventos de maneira assíncrona. O consumidor precisa ser informado quando um evento ocorre e ele pode processar ou apenas ser afetado pelo evento.

O Apache Kafka é uma plataforma de transmissão de dados distribuída muito usada para o processamento de eventos. Ele é capaz de realizar publicações, subscrições, armazenamento e processamento de fluxos de eventos em tempo real.

2.5. **Arquitetura Serverless**

A Arquitetura Serverless (sem servidor) permite criar e executar aplicações e serviços em nuvem, sem a necessidade do gerenciamento de infraestrutura. A aplicação continua sendo executada em servidores, mas esses servidores são totalmente gerenciados pelo serviço em Nuvem, sem a necessidade de provisionar, escalar e manter servidores para executarem as nossas aplicações e serviços.

3. **Metodologias Ágeis**

Considerada um modelo alternativo à gestão de projetos tradicional, a Metodologia Ágil visa aperfeiçoar o processo de desenvolvimento de um produto ou serviço, tendo como objetivo final a realização de entregas com rapidez de acordo com as necessidades do cliente. Ela foi planejada com base nas necessidades dos desenvolvedores de software.

A Metodologia Ágil busca otimizar fluxos de trabalho, melhorar a produtividade de projetos e elevar as perspectivas de sucesso do negócio. Sendo aplicada em todo o ciclo de vida do projeto, desde a sua concepção até a entrega do produto final.

3.1. **Manifesto Ágil**

O que se conhece hoje por Metodologias Ágeis está descrito no Manifesto Ágil para Desenvolvimento de Software. Assinado em 2001, em Utah, nos EUA, o material foi construído por 17 desenvolvedores que testavam abordagens diferentes, mas com fundamentos iguais. Em suma, defendem o planejamento adaptativo, times multidisciplinares e com autonomia, melhoria contínua e o desenvolvimento evolucionário. Abaixo estão os 4 fundamentos-chave desse documento:

1) Indivíduos e interações acima de processos e ferramentas
2) Software funcionando acima de documentação abrangente
3) Colaboração com o consumidor/cliente acima de negociação de contratos
4) Resposta às transformações/mudanças, mais do que seguir um plano

Além disso, o Manifesto é comporto por 12 princípios, com o intuito de otimizar o trabalho das equipes e aumentar os resultados gerados para o cliente:

1) Ter como prioridade a satisfação do cliente por meio de entregas de valor contínuas e rápidas.

2) Ser receptivo a alterações nos requisitos em qualquer fase do processo. Aliás, ambientes mutáveis são empregados em toda etapa do projeto para entregar ao cliente vantagem competitiva.
3) Realizar entregas frequentes (do produto ou serviço) no menor período de tempo possível;
4) Manter a colaboração das partes envolvidas em todo o projeto, diariamente;
5) Fornecer o ambiente, as ferramentas e o suporte necessários aos indivíduos do projeto, além de acreditar neles para realizar as atividades.
6) Estimular a comunicação pessoal, que transmite as informações necessárias ao time de colaboradores, sendo o meio mais eficiente. Nesse ponto, atenção especial para reuniões presenciais, consideradas mais eficazes para o sucesso do projeto.
7) Um produto final de trabalho corresponde à medida final do êxito. No caso da tecnologia, a medida primária de progresso consiste no software em funcionamento.
8) Os profissionais envolvidos no processo precisam manter um ritmo constante, de modo indefinido, pois fluxos ágeis estimulam um desenvolvimento sustentável. Da mesma maneira, o desenvolvimento sustentável é feito por intermédio de processos ágeis, por meio dos quais as partes interessadas conseguem manter um ritmo contínuo e cíclico.
9) Manter atenção frequente à excelência de design e técnica eleva ou aprimora a agilidade.
10) Eliminar o máximo de esforços que não geram valor ao produto, pois a simplicidade é fundamental.
11) Equipes auto-organizáveis propiciam os melhores designs e arquiteturas, além de atenderem aos requisitos do projeto.
12) Por meio de intervalos regulares, o time de colaboradores do projeto reflete sobre como melhorar a sua eficiência e eficácia para otimizar o seu comportamento.

Referência:
https://agilemanifesto.org/iso/ptbr/manifesto.html

3.2. **Vantagens da Metodologia Ágil**

Além de garantir a eficácia no desenvolvimento e gerenciamento de projetos, a implementação de metodologias ágeis na gestão de projetos, pode trazer uma série de benefícios tanto para as empresas, como para todos os profissionais envolvidos:

Aumento da produtividade
Com um processo produtivo mais rápido e eficaz, a produtividade do seu time aumenta.
O trabalho é focado nos componentes e tarefas que fazem a diferença para o cliente, agilizando as entregas.

Melhorias na comunicação
As metodologias ágeis pregam que o fluxo de trabalho seja iterativo, tanto em relação às equipes e seus gestores como da empresa com os clientes. Ou seja, a comunicação é mais transparente, direta e eficaz em todas as partes do projeto.

Mais qualidade
Assim, o produto final liberado ao mercado é, em teoria (e na prática também, se a metodologia ágil for realmente bem-sucedida), é a entrega ideal. Trata-se de uma forma de colocar no mercado uma solução que você sabe que é a ideal para seus clientes, que vai sanar suas dores e necessidades.

Integração entre pessoas
De maneira geral, a metodologia ágil cria muitas oportunidades para a comunicação entre as pessoas envolvidas no projeto. Desse modo, há um alto nível de colaboração entre todas as partes, o que torna

mais transparente a visão do cliente e suas expectativas, bem como as possibilidades do time de desenvolvimento.

Respostas rápidas para imprevistos e alterações

Com a estrutura ágil, é possível redefinir prioridades e realizar mudanças de maneira mais flexível do que em estruturas tradicionais de projeto. Assim, as mudanças podem ser elencadas pelo gestor e adicionadas nas próximas sprints, de maneira integrada ao desenvolvimento, impactando-o positivamente.

3.3. **Principais Metodologias Ágeis**

Conheça as principais metodologias ágeis utilizadas no mercado e que utilizaremos no nosso treinamento:

Kanban, termo de origem japonesa que significa literalmente “cartão” ou “sinalização”.
Seu conceito está relacionado ao uso de cartões post-it, luzes e caixas vazias, para indicar o status de transportes ou fluxos de produção em companhias de fabricação em série.

Scrum consiste em uma metodologia ágil para planejamento e gerenciamento de projetos (especialmente de software). Nele, cada projeto é segmentado em ciclos, geralmente mensais, conhecidos como sprints, que consistem em um time box (caixa de tempo) ou um intervalo em que um conjunto de atividades deve ser realizado.

4. **Visão geral das tecnologias abordadas neste treinamento**

Neste treinamento, além das tecnologias que aprenderemos relacionadas as stacks front-end e back-end, vamos conhecer novas tecnologias e aprender a utilizá-las em nossos projetos.

4.1. **Fundamentos do versionamento (utilizando o Git)**

O controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo, possibilitando a recuperação de versões específicas posteriormente. Ele permite reverter arquivos selecionados para um estado anterior, reverter todo o projeto para um estado anterior, comparar alterações ao longo do tempo, ver quem modificou pela última vez algo que pode estar causando um problema, quem introduziu um problema, quando e muito mais.

Usar um sistema de controle de versão também significa que, se você perder arquivos ou alterar algo que não deveria e prejudicar o funcionamento do seu projeto local, poderá recuperar facilmente os arquivos do repositório principal, retornando ao estado anterior da sua alteração.

O Git é um sistema de controle de versão distribuído gratuito e de código aberto, com ele podemos desenvolver projetos onde diversas pessoas podem contribuir simultaneamente, editando e criando novos arquivos, sem o risco de sobrescrever as alterações.

Com o Git é possível rastrear e monitorar todas as alterações realizadas nos códigos fontes, facilitando a coordenação do trabalho em equipe, a organização e a realização das entregas dos artefatos do projeto, disponibilizando o código fonte do projeto, testado e funcionando no ambiente de produção.

Referência:

https://git-scm.com/book/en/v2

Atividades:
1) Caso ainda não possua uma conta pessoal no GitHub, crie a sua conta pessoal para subir os projetos que você vai criar durante o nosso treinamento, ampliando o seu portfólio de projetos pessoais!
https://github.com

2) Instale a ferramenta para o gerenciamento dos repositórios dos projetos na sua estação de trabalho, utilizaremos o GitHub Desktop:
https://desktop.github.com

4.2. **Automatizando tarefas com Maven e Gradle**

Maven e Gradle são ferramentas de automação de builds e controle de dependências, utilizadas em Projetos Java e outros projetos baseados em JVM para gerenciar dependências, criar e empacotar aplicativos e executar outras tarefas relacionadas.

O Maven é uma ferramenta mais antiga e estabelecida, surgiu em 2005 e usa um arquivo de configuração baseado em XML para gerenciar os projetos. Ele utiliza plugins para executar várias tarefas, como compilar código, executar testes e empacotar aplicativos. Ele também possui um grande repositório central para download de dependências.

Por outro lado, o Gradle é uma ferramenta mais recente e mais robusta, usa uma linguagem específica de domínio (DSL) baseada em Groovy para configurar os projetos, os scripts do Gradle são declarativos, de fácil leitura, bem mais legível e o tamanho do script é bem menor. Ele tem uma abordagem de configuração de compilação mais flexível e personalizável e suporta compilações incrementais e execução paralela. O Gradle usa uma arquitetura de plugins semelhante ao Maven e também possui vários plugins nativos.

Em geral, ambas as ferramentas são capazes e amplamente utilizadas, a escolha entre as duas depende das necessidades e preferências específicas da equipe, devemos sempre nos atentar e seguir os padrões adotados pela empresa.

Na verdade, o Maven e o Gradle não são rivais, o Gradle veio para melhorar o que já era bom!

Referências:
https://maven.apache.org/
https://docs.gradle.org/current/userguide/userguide.html

4.3. **IDEs (Ferramentas para o desenvolvimento de softwares)**

Um ambiente de desenvolvimento integrado (IDE) é uma aplicação de software que ajuda os programadores a desenvolver código de software de maneira eficiente. Ele aumenta a produtividade do desenvolvedor, combinando recursos como edição, compilação, teste e empacotamento de software em uma aplicação visual e fácil de usar.

Por que as IDEs são importantes?
Você pode usar qualquer editor de texto para escrever código. No entanto, a maioria dos ambientes de desenvolvimento integrado (IDEs) incluem funcionalidades que tornam o processo de desenvolvimento de software muito mais eficiente, onde os desenvolvedores podem começar a programar novas aplicações

rapidamente, sem precisar integrar e configurar manualmente diferentes softwares, se concentrando apenas na IDE.

A escolha da IDE depende muito da linguagem ou área de atuação, sendo bem mais pessoal hoje em dia, onde as empresas já não obrigam a utilizar uma determinada IDE. Para o nosso treinamento que é voltado para a linguagem Java, além do IntelliJ (IntelliJ IDEA Community Edition) que vamos adotar como a nossa IDE padrão, o Eclipse também é bastante utilizado pelos desenvolvedores.

Download do IntelliJ IDEA Community Edition:
https://www.jetbrains.com/pt-br/idea/download