

ARQUITETURA SERVER-SIDE

Cassio Trindade e Miguel Gomes Xavier



Qualidade não é opcional, temos que saber
tratá-la dentro do nosso desenvolvimento de
sistemas.



Daniel Wildt

Conheça o livro da disciplina

CONHEÇA SEUS PROFESSORES

3

Conheça os professores da disciplina.

EMENTA DA DISCIPLINA

4

Veja a descrição da ementa da disciplina.

BIBLIOGRAFIA DA DISCIPLINA

5

Veja as referências principais de leitura da disciplina.

O QUE COMPÕE O MAPA DA AULA?

6

Confira como funciona o mapa da aula.

MAPA DA AULA

7

Veja as principais ideias e ensinamentos vistos ao longo da aula.

RESUMO DA DISCIPLINA

21

Relembre os principais conceitos da disciplina.

AVALIAÇÃO

22

Veja as informações sobre o teste da disciplina.

Conheça seus professores



CASSIO TRINDADE

Professor Convidado

Profissional da área de TI, trabalhando há mais de uma década com a formação de profissionais, dando aulas no Instituto Federal do Rio Grande do Sul, na Faculdade Dom Bosco, Universidade Luterana do Brasil (ULBRA), Pontifícia Universidade Católica do RS (PUCRS) e na TargetTrust. Atualmente atuando como Arquiteto de Software na PUCRS, sendo responsável pela condução e elaboração de mais de 90 projetos diretamente com alunos do curso de Engenharia de Software, trabalhando com as mais variadas tecnologias. Mais de 30 anos de experiência nas áreas de desenvolvimento de software, aplicativos para celulares e sistemas corporativos e para internet desde projetos de e-commerce para o Sonae Portugal e site de classificados digitais do Grupo RBS a dezenas de aplicativos mobiles.

MIGUEL GOMES XAVIER

Professor PUCRS

Possui mestrado em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul e está cursando doutorado em Ciência da Computação na mesma instituição, atuando principalmente nas áreas de alto desempenho, sistemas distribuídos, virtualização e cloud computing. Atualmente participa de projetos de pesquisa em cooperação com diferentes universidades envolvendo gerência de recursos em arquiteturas de alto desempenho. Tem participado de projetos de análise de dados (BigData), realizando contribuições científicas em prol do avanço da área na indústria e na academia.



Ementa da Disciplina

Estudo sobre Arquitetura cliente-servidor para aplicações web. Introdução aos frameworks MVC server-side: Node.js, Express, Nestjs. Estudo de programação assíncrona e programação reativa. Desenvolvimento de aplicações web com o conceito de uso de serviços.

Bibliografia da Disciplina

As publicações destacadas têm acesso gratuito.

Bibliografia básica

PESSMAN Roger S; Maxim, Bruce R. Engenharia de software: uma abordagem profissional. 9a. Ed. Porto Alegre: AMGH, 2021.

BROWN, Ethan. Programação web com Node e Express: beneficiando-se da Stack Javascript. São Paulo: Novatec, 2020.

IHRIG, Colin J. Pro Node.js para desenvolvedores. Ciência Moderna, 2020.

Bibliografia complementar

[WHATWG. HTML living standard. WHATWG Web Hypertext Application Technology Working Group, 2021.](#)

ALVES, William Pereira. Projeto de sistemas web: conceitos, estruturas, criação de banco de dados e ferramentas de desenvolvimento. São Paulo: Érica, 2015.

[GRONER, Loiane. Learning JavaScript Data Structures and Algorithms. Third Edition. Birmingham: Packt, 2018.](#)

OLIVEIRA, C. L. V.; ZANETTI, H. A. P. Javascript Descomplicado: programação para web, IOT e dispositivos móveis. São Paulo: Érica, 2020.

GALLOTI, Giocondo M. Antonio. Arquitetura de Software. São Paulo: Pearson do Brasil, 2016.

O que compõe o Mapa da Aula?

MAPA DA AULA

São os capítulos da aula, demarcam momentos importantes da disciplina, servindo como o norte para o seu aprendizado.



EXERCÍCIOS DE FIXAÇÃO

Questões objetivas que buscam reforçar pontos centrais da disciplina, aproximando você do conteúdo de forma prática e exercitando a reflexão sobre os temas discutidos. Na versão online, você pode clicar nas alternativas.



PALAVRAS-CHAVE

Conceituação de termos técnicos, expressões, siglas e palavras específicas do campo da disciplina citados durante a videoaula.



VÍDEOS

Assista novamente aos conteúdos expostos pelos professores em vídeo. Aqui você também poderá encontrar vídeos mencionados em sala de aula.



PERSONALIDADES

Apresentação de figuras públicas e profissionais de referência mencionados pelo(a) professor(a).



LEITURAS INDICADAS

A jornada de aprendizagem não termina ao fim de uma disciplina. Ela segue até onde a sua curiosidade alcança. Aqui você encontra uma lista de indicações de leitura. São artigos e livros sobre temas abordados em aula.



FUNDAMENTOS

Conteúdos essenciais sem os quais você pode ter dificuldade em compreender a matéria. Especialmente importante para alunos de outras áreas, ou que precisam relembrar assuntos e conceitos. Se você estiver por dentro dos conceitos básicos dessa disciplina, pode tranquilamente pular os fundamentos.

CURIOSIDADES

Fatos e informações que dizem respeito a conteúdos da disciplina.

DESTAQUES

Frases dos professores que resumem sua visão sobre um assunto ou situação.

ENTRETENIMENTO

Inserções de conteúdos para tornar a sua experiência mais agradável e significar o conhecimento da aula.

CASE

Neste item, você relembra o case analisado em aula pelo professor.

MOMENTO DINÂMICA

Aqui você encontra a descrição detalhada da dinâmica realizada pelo professor.

Mapa da Aula

Os tempos marcam os principais momentos das videoaulas.

AULA 1 • PARTE 1

Introdução

A Arquitetura Server-Side é toda a parte da solução de construção de software que é realizada no lado do servidor. Para isso funcionar bem é necessário uma arquitetura muito bem definida. Nesta aula iremos conhecer as principais características da Arquitetura Server-Side, focando na arquitetura mais utilizada e nas tecnologias de construção mais novas.

“Para nós, 30 anos é muita coisa na tecnologia da informação.”

“Interface é a ligação entre uma coisa e outra.”

“É muito comum nós começarmos a desenvolver software sem pensar na arquitetura antes, e isso nos gera muito problema depois.”

01:10

03:45

Model-View-Controller (MVC)

Atualmente, ninguém constrói um software sem utilizar algum tipo de framework de design pattern. É fundamental entender esses conceitos para poder criar softwares profissionais e de qualidade. Antigamente os softwares eram construídos com um único código, de forma que a manutenção era extremamente complexa. Com a intenção de tornar esses softwares mais fáceis de trabalhar, um dos primeiros frameworks modernos criados foi o Model-View-Controller. O MVC separa os códigos em, pelo menos, três grandes agrupamentos:

Model: esta é a parte do nosso aplicativo que gerencia o banco de dados e todas as operações relacionadas aos dados. Ele cuida do armazenamento, recuperação e manipulação dos dados essenciais para o funcionamento da aplicação.

View: a “visão” engloba tudo o que é visível para o usuário. Em termos simples, são as páginas e elementos visuais que são apresentados ao cliente. A visão é responsável por garantir uma experiência agradável ao usuário, cuidando da forma como os dados são apresentados.

Controller: o “controlador” é o cérebro da nossa aplicação. Ele contém a lógica que coordena a interação entre o

04:50

06:44

08:01

PALAVRA-CHAVE

JavaScript Object Notation (JSON):

É um formato de dados leve e de fácil leitura utilizado para troca de informações entre sistemas computacionais.



09:15

modelo e a visão. No controlador, chamamos os modelos para obter dados, processamos esses dados e os disponibilizamos para as visões para que sejam entregues aos usuários. Além disso, o controlador é o local onde desenvolvemos funcionalidades, as expandimos e realizamos a manutenção da aplicação.

Os principais benefícios da utilização do MVC são:

- Separação de responsabilidade;
- Reutilização de código;
- Facilidade de manutenção;
- Escalabilidade;
- Facilidade de colaboração;
- Facilidade de migração;
- Testabilidade e Documentação.

Programação assíncrona

É importante conhecermos dois paradigmas de programação: a programação síncrona e a programação reativa.

A programação assíncrona é um paradigma de programação que permite que uma aplicação execute tarefas em paralelo, em vez de sequencialmente, o que é comum em programação síncrona. Isso significa que, em vez de esperar que uma tarefa seja concluída antes de iniciar outra, as tarefas podem ser executadas simultaneamente ou em segundo plano, permitindo que o programa continue sua execução principal sem bloquear.



28:05

EXERCÍCIO DE FIXAÇÃO

Qual é a função principal do padrão MVC?



“ Node é uma linguagem, por padrão, assíncrona. ”



35:26

AULA 1 • PARTE 2

Programação reativa

A programação reativa é um paradigma de programação que se concentra na criação de sistemas e aplicativos que respondem automaticamente a mudanças de estado e eventos. Ela é particularmente útil para desenvolver aplicativos em tempo real, como aplicativos da web que exigem atualizações em tempo real com base em ações do usuário ou em eventos de sistema.

Os aplicativos cada vez mais tendem a atuar conforme as necessidades do usuário. Para isso, fazemos uso de algumas tecnologias na programação reativa:

- Observáveis e observadores (observable/observer);
- Bibliotecas de programação reativa;
- Fluxos de eventos;
- Assinatura e cancelamento;
- Tratamento de erros;
- Aplicações em tempo real;
- Integração com bancos de dados;
- Aplicações assíncronas de alto desempenho concorrentes;
- Testes unitários;
- Padrões de design reativos;
- Operadores reativos.

05:37

00:25



Node.js

Durante muito tempo não foi possível fazer outras coisas além do desenvolvimento frontend. Surgiu então um framework baseado em linguagem JavaScript chamado Node, que foi uma revolução. A partir desse momento foi possível ter uma aplicação JavaScript do lado do servidor. O Node nasceu com o servidor HTTP embutido e sofreu uma evolução com o Express, que é um servidor mais requintado HTTP.

O Node.Js é um interpretador JavaScript do lado do servidor que altera a noção de como um servidor deveria funcionar. Possibilita que um programador crie aplicativos altamente escaláveis e que se escreva códigos que manipulam dezenas de milhares de conexões simultâneas. É uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.

Características do Node.Js:

- Não bloqueante (non-blocking-thread);
- Javascript Engine V8 (altamente escalável);
- Single-thread (cada aplicação terá instância de um único processo);
- Orientado a eventos (a mesma lógica a de orientação de eventos do Javascript client-sided).

PALAVRA-CHAVE

Node Package Manager (npm): É uma ferramenta do Node.js para o gerenciamento de pacotes. Essa utilidade auxilia na instalação e desinstalação de pacotes, gerenciamento das versões e gerenciamento de dependências necessárias para executar um projeto.



12:25

PALAVRA-CHAVE

Visual Studio Code: É um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS.

“ O que vocês precisam para programar em Node é conhecer JavaScript. **”**



23:59

“ Toda forma que eu conheço e utilizo para programar em JavaScript, eu programo em Node. Js. **”**

Request e response

É muito importante entendermos como ocorre a troca de informações entre o servidor e o cliente. Do lado do servidor há aplicação Node.js, mas do lado do cliente a linguagem da aplicação não é importante desde que se comunique utilizando os conceitos do HTTP. O protocolo HTTP é baseado em troca de informações através de **request** e **response**.

Representational State Transfer (Rest) é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de web services. O Rest permite processar as requisições HTTP através de verbos (GET, POST, DELETE, PUT, PATCH).



33:08



EXERCÍCIO DE FIXAÇÃO

O protocolo HTTP é baseado em troca de informações através de:

AULA 1 • PARTE 3

Server com HTTP

Neste momento da aula, o professor faz uma demonstração prática de construção de um servidor com HTTP. É utilizada a extensão Express, um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.



00:25

EXERCÍCIO DE FIXAÇÃO

O que caracteriza a programação reativa?



“Node modules é uma das pastas mais importantes quando a gente está falando de Node. **”**



15:37

AULA 2 • PARTE 1

API Rest com Postman

O professor inicia a aula testando com o Postman a API Rest criada anteriormente. O Postman é uma ferramenta utilizada por desenvolvedores e equipes de desenvolvimento de software para testar, documentar e colaborar em APIs. Ele oferece uma interface que permite enviar solicitações HTTP para APIs e receber as respostas correspondentes.

O Postman pode ser baixado [neste link](#), além de também poder ser utilizado online.



01:09

01:34



“Existem várias outras ferramentas, eu gosto mais do Postman. **”**

20:03



“Uma outra facilidade muito legal do Postman é que ele utiliza variáveis de ambiente. **”**

“ Para cada um dos sistemas que eu trabalho, eu tenho um conjunto de variáveis diferentes. **”**

22:45

36:12

API Rest com Express - Parte I

Neste momento da aula, o professor inicia a demonstração prática da criação de uma aplicação mais robusta utilizando o Node e o Express.

AULA 2 • PARTE 2

API Rest com Express - Parte II

Continuação da demonstração prática da criação da aplicação Postcard-collection com o Node e Express, relacionando com cada um dos benefícios do MVC (separação de responsabilidade, reutilização de código, facilidade de manutenção, escalabilidade, facilidade de colaboração, facilidade de migração e testabilidade e documentação).

00:25

18:54

“ Neste primeiro código que a gente trabalhou a gente não tem tanto essas separações [...], mas a gente consegue ver um reaproveitamento. **”**

“ Essa nossa lógica aqui é muito simples porque a gente só está buscando direto em um arquivo JSON. **”**

34:33

AULA 2 • PARTE 3

API Rest com Express - Parte III

Continuação da demonstração prática da criação da aplicação Postcard-collection com o Node e Express, relacionando com cada um dos benefícios do MVC (separação de responsabilidade, reutilização de código, facilidade de manutenção, escalabilidade, facilidade de colaboração, facilidade de migração e testabilidade e documentação).



00:25

36:12



Implementando o MongoDB - Parte I

Depois de separar as responsabilidades do Postcard-collection a próxima alteração a ser feita é não utilizar mais o arquivo JSON. É possível utilizar uma estrutura de banco de dados para facilitar as três grandes áreas do MVC, dessa forma o model será tratado pelo MongoDB.

PALAVRA-CHAVE

MongoDB: É um banco de dados não-relacional orientado a documentos, ou seja, que não se utiliza de tabelas e colunas pré-definidas.



36:45



EXERCÍCIO DE FIXAÇÃO

É um software de código aberto usado para implantar aplicativos dentro de containers virtuais.



38:01

“ Vocês têm que ter o banco [de dados] instalado, porque se vocês não tiverem o banco instalado de alguma forma ele não vai funcionar. **”**

PALAVRA-CHAVE

Docker: É um software de código aberto usado para implantar aplicativos dentro de containers virtuais.



38:26

AULA 2 • PARTE 4

Implementando o MongoDB - Parte II

00:25

O professor continua a demonstração prática de como implementar o banco de dados MongoDB na aplicação Postcard-collection.

Nest.Js

O Nest.Js é um framework Node.js progressivo para construir aplicativos do lado do servidor eficientes, confiáveis e escaláveis. É uma plataforma recente, sendo uma das principais soluções que facilita que as páginas tenham relevância em buscas. Tende a trabalhar em um único servidor o lado cliente e o lado do servidor.

PALAVRA-CHAVE

Search Engine Optimization (SEO):

É um conjunto de técnicas que tem como objetivo posicionar uma ou mais páginas de destino entre os melhores resultados dos mecanismos de busca.

25:42

Para compreendermos melhor a utilização do Nest.Js, o professor faz uma demonstração prática na aplicação Card-collection.

PALAVRA-CHAVE

CRUD: É o acrônimo para Create (criar), Read (ler), Update (atualizar) e Delete (apagar). São as quatro operações básicas do desenvolvimento de uma aplicação, sendo utilizadas em bases de dados relacionais fornecidas aos utilizadores do sistema.

31:18

AULA 3 • PARTE 1

Introdução

Miguel Xavier apresenta uma introdução ao conteúdo que está sendo abordado nesta disciplina. Xavier discorre sobre as mudanças tecnológicas que ocorreram ao longo do tempo nesta questão de front-ends mais pesados e maior tempo de carregamento para os usuários. A renderização do lado do servidor está ganhando cada vez mais tração.



04:48

05:12



“O que a indústria sempre buscou através de ensinamentos e de descobertas realizadas na academia foi sempre trazer a melhor experiência para o usuário, seja em termos de velocidade, no acesso, ou seja em termos de acessibilidade. **”**



09:55



Renderizar, do lado do servidor é realizar essa tarefa toda do lado do servidor e o cliente agir somente como um cliente que faz uma solicitação de um dado na rede exibe esse dado no browser. **”**

EXERCÍCIO DE FIXAÇÃO

O protocolo HTTP é baseado em troca de informações através de:



13:40

**SSR**

SSR, ou Server-Side Rendering, é uma técnica usada no desenvolvimento web onde o conteúdo da página é processado e gerado no servidor, em vez de no navegador do usuário. Isso melhora o desempenho, a indexação por mecanismos de busca (SEO) e a experiência inicial do usuário, especialmente em conexões lentas. O professor apresenta alguns exemplos de páginas web com pré-renderização através do SSR.

Pré-renderização

Miguel cita alguns exemplos práticos sobre a pré-renderização. Ele discorre que para a técnica de pré-renderização funcionar é necessário ter um proxy ou algo que redirecione o usuário para o arquivo correto. Segundo Miguel, o sistema de rotas para as diferentes páginas na web se dá através de um roteamento de forma integrada que irá acontecer do lado do cliente, então, através deste ponto ele começa a fazer o roteamento em função daquilo que o usuário solicita.

14:32



14:32



26:48

28:18



Então, se o usuário solicita o acesso a uma página de login, a uma página de perfil de usuário, coisas do tipo, isso é uma requisição nova que acontece para o lado do servidor.

**AULA 3 • PARTE 2****Pré-renderização**

Xavier dá continuidade na explicação sobre o conceito da pré-renderização, apresentando exemplos de como ela funciona nas páginas da web para o usuário e, também, para o servidor. Segundo Miguel, existem casos em que a preocupação é a segurança, assim, trazer os dados para o lado do cliente não é interessante, como, por exemplo, no caso das telas de login, que será renderizada sempre que o usuário tentar acessar uma rota, ou URL, cujo acesso é negado.

00:01



02:53



Não adianta ter um sistema que mantém o usuário ativo por uma boa experiência, mas tem que ter a funcionalidade, de fato entregando o que ela se propõe a fazer.



- “** Então, a partir do momento em que o lado do cliente identifica esse código de não autorização, ele mesmo já vai tratar de redirecionar o usuário para a tela de login. **”**
- 14:02 05:45
- “** Independente, no caso do cliente, de qual framework, no final do dia esses frameworks, não são linguagem de programação, a linguagem aqui é Javascript. **”**
- 16:13
- “** O propósito do node é fazer com que o Javascript possa ser utilizado no back-end fora do navegador. **”**
- 31:32 16:13
- JS Back-end development**
- Xavier explica sobre o lado do servidor, no intuito de prover todos os artefatos que são necessários para que o SSR funcione, para que o navegador consiga fazer a renderização. O professor complementa também sobre o nestJS como um framework node.js para construção de aplicação back-end, abordando a produtividade, a performance e a escabilidade.
- “** Basta que um dos componentes não consiga entregar a performance para que o sistema, como um todo, fracasse. **”**
- 34:07
- “** 35:41
- Nest**
- O docente explica sobre o framework nest, um framework opinado, ou seja, construído para que o programador seja direcionado para um caminho que siga as boas práticas em definição de arquitetura, por isso ele não dá muita liberdade para decidir como o programador vai arquitetar seu software.
- AULA 3 • PARTE 3**

- REST API**
- Nesta parte da aula, o professor Xavier disserta sobre a web service REST API, explicando que escolheu falar sobre ele de início porque o REST API é um padrão que vem ganhando muita força, pois seu objetivo não é fornecer códigos, mas sim fornecer os dados. Xavier apresenta exemplos práticos de código.
- 00:01
- 03:07 00:01
- “** Então, o REST API está associado a um controller. **”**



EXERCÍCIO DE FIXAÇÃO



De acordo com Xavier, o REST API está associado a um:

AULA 3 • PARTE 4

Middlewares

Xavier dá seguimento ao conteúdo sobre os middlewares apresentando uma série de exemplos práticos de classes e, também, de métodos de suas aplicações. O professor alerta sobre a importância de estar atento à necessidade de sempre definir, dentro do estilo arquitetural que se está construindo, quais são os componentes que são passíveis de lançar e exceções.



00:01

13:07



Pipes

O Pipe é um mecanismo utilizado em arquitetura de software, cujo seu propósito é criar uma forma de fazer a conexão de uma entrada com uma saída, no caso, de um programa para outro. De acordo com o professor, o Pipe é muito utilizado em streaming de dados quando se quer, ao longo do fluxo de dados, começar a fazer transformações, pegando algo que chega na entrada e jogando na saída de forma diferente.



20:32

“Eu não posso, dentro do meu sistema, deixar entrar alguma coisa que eu não conheço.**”**

21:06



Testing

Xavier explica que é possível fazer algumas atividades e tarefas de maneira simples e automatizada por meio de algumas primitivas mais voltadas para testes unitários. Ele apresenta mais algumas classes exemplificando de forma prática suas aplicações.



33:08

“O caching é uma estratégia muito forte e muito utilizada para sistemas que são críticos de performance.**”**



34:12

“Assim como o nest está para back-end, o next está para front-end.**”**

EXERCÍCIO DE FIXAÇÃO

Qual é o propósito do Pipe, de acordo com o professor Xavier?



Resumo da disciplina

Veja, nesta página, um resumo dos principais conceitos vistos ao longo da disciplina.

AULA 1

O MVC separa os códigos em, pelo menos, três grandes agrupamentos



Os aplicativos cada vez mais tendem a atuar conforme as necessidades do usuário.

A extensão Express fornece um conjunto robusto de recursos para aplicativos web e móvel.



AULA 2

Para facilitar as três grandes áreas do MVC, é possível utilizar o MongoDB.



O Postman é uma ferramenta utilizada por desenvolvedores para testar, documentar e colaborar em APIs.



O Nest.Js Tende a trabalhar em um único servidor o lado cliente e o lado do servidor.



AULA 3

Para que a técnica de pré-renderização funcione é necessário ter um proxy, ou algo que redirecione o usuário para o arquivo correto.



O middleware faz a intermediação de acesso entre duas entidades e é muito utilizado em cloud computing,



O pipe é um mecanismo utilizado em arquitetura de software, cujo seu propósito é criar uma forma de fazer a conexão de uma entrada com uma saída.

Avaliação

Veja as instruções para realizar a avaliação da disciplina.

Já está disponível o teste online da disciplina. O prazo para realização é de **dois meses a partir da data de lançamento das aulas**.

Lembre-se que cada disciplina possui uma avaliação online.
A nota mínima para aprovação é 6.

Fique tranquilo! Caso você perca o prazo do teste online, ficará aberto o teste de recuperação, que pode ser realizado até o final do seu curso. A única diferença é que a nota máxima atribuída na recuperação é 8.

