Universidade Federal do Amazonas
Instituto de Computação

Programação para a Web – 2025/01
Segundo Trabalho Prático

Introdução

O segundo trabalho prático da disciplina de Programação Web consiste no desenvolvimento de uma aplicação Web que permitirá aos usuários jogar o jogo Space Shooter, previamente desenvolvido no primeiro trabalho prático. A aplicação deverá ser implementada utilizando o framework Express com TypeScript, em conjunto com o ORM Prisma para acesso ao banco de dados. A aplicação deve permitir que qualquer usuário possa criar uma conta, realizar login e acessar a página onde será possível jogar o Space Shooter. Além disso, deverá ser implementada uma página de ranking, que exibirá a lista dos usuários com as maiores pontuações no jogo.

Etapas do Trabalho Prático

Boa parte do segundo trabalho prático consiste na resolução de todos os exercícios apresentados nos slides sobre o framework Express. Cada exercício proposto representa, na prática, uma etapa no desenvolvimento da aplicação que compõe o segundo trabalho prático. No entanto, além dos exercícios dos slides, o trabalho inclui outras tarefas adicionais que não serão abordadas durante as aulas, sendo responsabilidade dos alunos implementá-las de forma autônoma.

IMPORTANTE: o trabalho terá valor total de 10,0 pontos, sendo 5,0 pontos atribuídos aos exercícios resolvidos pelo professor durante as aulas e 5,0 pontos correspondentes aos exercícios que não foram resolvidos em sala. Os dois conjuntos de exercícios estão listados abaixo.

#1 (Resolvido pelo professor): Criar um projeto de uma aplicação Web utilizando o framework Express com TypeScript. É necessário instalar e usar os pacotes **dotenv** e **nodemon**, além de configurar no arquivo package.json três scripts: **start**, que deve executar a aplicação em ambiente de desenvolvimento utilizando o nodemon; **deploy**, que deve gerar a versão transpilada do código utilizando o compilador do TypeScript, armazenando os arquivos no diretório build; e **start:prod**, que deve executar a versão transpilada da aplicação localizada no diretório build, simulando assim o ambiente de produção.

#2 (Resolvido pelo professor): Desenvolver a página About, que deverá apresentar informações básicas e gerais sobre o jogo Space Shooter (pode ser um texto simples, obtidas de qualquer fonte disponível na Internet). Essa página deverá ter uma ou mais imagens, que deverão ser disponibilizadas no diretório **/public/img** da aplicação. O **middleware de arquivos estáticos** do Express deverá ser usado para mostrar as imagens na aplicação. A página Sobre deverá estar acessível na rota /about e utilizar a engine de templates Handlebars para sua renderização.

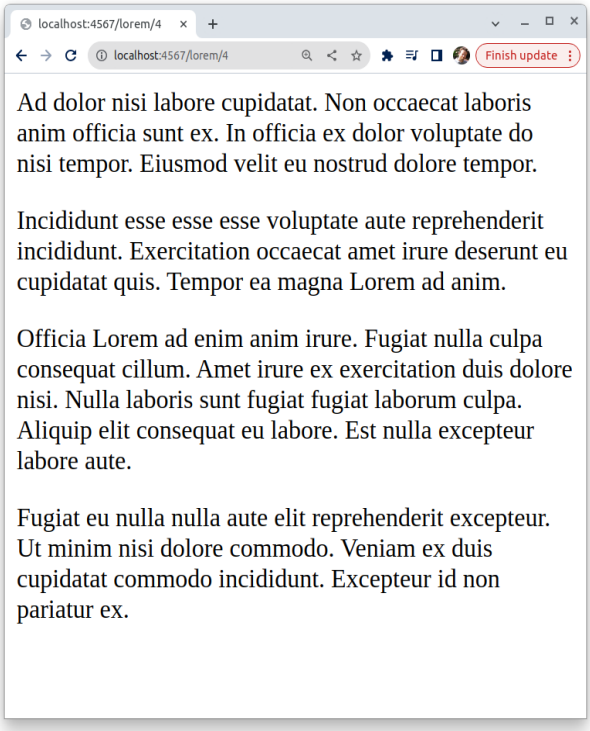
#3 (Resolvido pelo professor): Criar um arquivo **.env** (não versionável) e uma cópia desse arquivo **.env.example** (versionável). Criar um mecanismo de validação das variáveis de

ambiente definidas no arquivo **.env**. Esse mecanismo deverá ser baseado no pacode **envalid**, apresentado durante as aulas. O código de validação deverá ser implementado no arquivo **src/utils/validateEnv.ts**, que deverá ser importado em **src/index.ts** para validar as variáveis de ambiente.

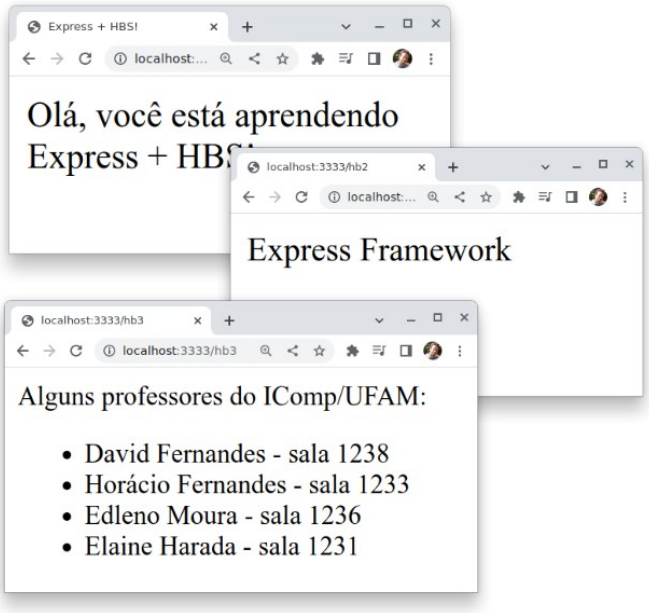
#4 (Resolvido pelo professor): Implementar um middleware chamado **logger**, localizado no diretório **src/middlewares**, cuja função é registrar os dados de acesso dos usuários em um arquivo de log. O caminho para esse arquivo deve ser definido por meio de uma variável chamada LOGS_PATH, declarada no arquivo .env, sendo obrigatória a validação dessa variável no arquivo **src/utils/validateEnv.ts**. O middleware deverá receber um parâmetro que define o formato do log, podendo ser **simples** ou **completo**. No formato simples, deverão ser registrados a hora do acesso, a URL acessada na aplicação e o método HTTP utilizado. No formato completo, além desses dados, também devem ser incluídos a versão do protocolo HTTP e o User-Agent do navegador utilizado pelo cliente.

#5 (Resolvido pelo professor): Colocar TODAS as rotas da aplicação em um arquivo separado, no path **src/router/router.ts**, seguindo as instruções dos slides.

#6 (NÃO resolvido pelo professor): Desenvolver uma rota **/lorem** que receba como parâmetro um valor inteiro qualquer. Como resultado, a rota deverá retornar um conteúdo HTML contendo uma quantidade de parágrafos igual ao valor informado. Dica: use o pacote lorem-ipsum.

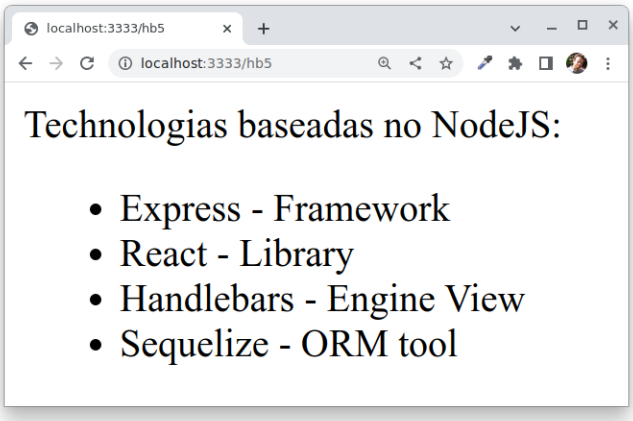


#7 (Resolvido pelo professor): Implementar as páginas das rotas **/hb1**, **/hb2** e **/hb3** mostradas nos slides. O conteúdo dessas páginas não precisa ser exatamente igual ao dos slides, mas a página /hb1 precisa imprimir o conteúdo de uma variável usando o Handlebars, e as páginas /hb2 e /hb3 precisam usar os comandos **#if** e **#each** do Handlebars, respectivamente.

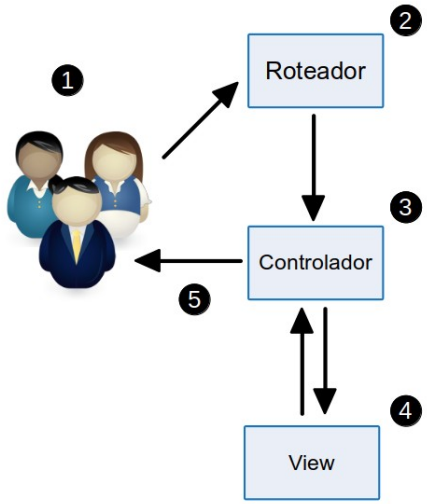


#8 (Resolvido pelo professor): Criar um helper para a engine de views Handlebars que recebe um vetor de tecnologias (com o mesmo formato do vetor technologies abaixo) e então retorna o código HTML contendo uma lista não ordenada com as tecnologias do vetor onde o valor de **poweredByNodejs** seja true. Crie uma página com a rota **/hb4** e use esse helper na página.

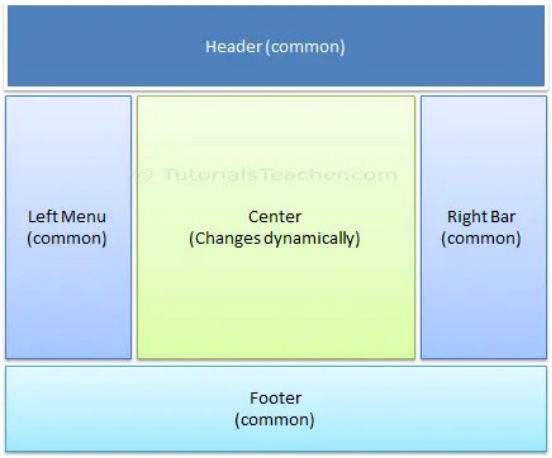
```
const technologies = [
  { name: 'Express', type: 'Framework', poweredByNodejs: true },
  { name: 'Laravel', type: 'Framework', poweredByNodejs: false },
  { name: 'React', type: 'Library', poweredByNodejs: true },
  { name: 'Handlebars', type: 'Engine View', poweredByNodejs: true },
  { name: 'Django', type: 'Framework', poweredByNodejs: false },
  { name: 'Docker', type: 'Virtualization', poweredByNodejs: false },
  { name: 'Sequelize', type: 'ORM tool', poweredByNodejs: true },
];
```



#9 (Resolvido pelo professor): Mover a aplicação para o MVC, seguindo os passos apresentados nos slides. Criar um controlador main com todas as rotas desenvolvidas nos exercícios anteriores. Note que a aplicação ainda não tem modelos e serviços, que somente serão criados nos próximos exercícios.

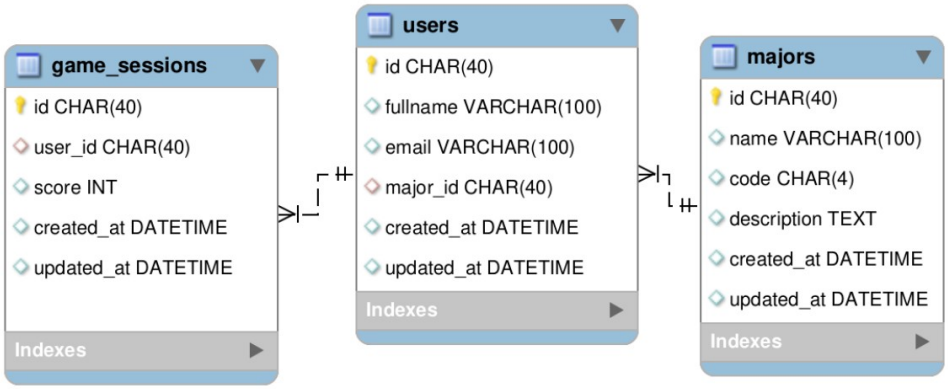


#10 (Resolvido pelo professor): Criar um layout Handlebars que contenha todo o conteúdo comum a todas as views da aplicação, incluindo as tags <html>, <head>, <body>, entre outras estruturas padrão de uma página HTML. Esse layout deverá ser utilizado de forma que cada view da aplicação contenha apenas o conteúdo específico da página que ela precisa renderizar, aproveitando assim a estrutura comum definida no layout.

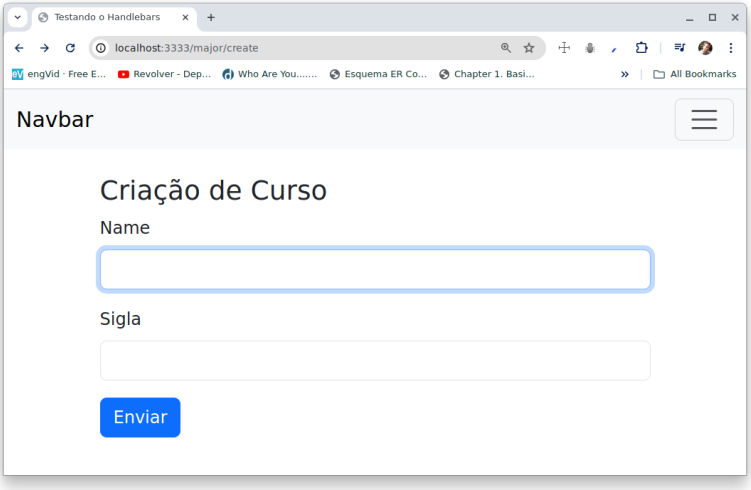


#11 (Resolvido pelo professor): Configurar o SASS na sua aplicação seguindo as instruções dos slides, e adicione uma cor no texto da view **/hb1** usando a técnica apresentada.

#12 (Resolvido pelo professor): Instalar e configurar o ORM Prisma na aplicação. Após isso, criar os modelos Major, User e GameSession. Rode a migration para que as tabelas sejam criadas no banco de dados.



#13 (Resolvido parcialmente pelo professor): Criar os CRUDs para os modelos Major e User. Cada CRUD deverá ter um controlador, um service, e um arquivo com os tipos/interfaces usados. Fazer a validação de todos os dados dos formulários usando o pacote Joi.



#14 (NÃO resolvido pelo professor): As páginas que exibem as listagens de Majors e Users, bem como as páginas de detalhes de um Major ou User específico, deverão disponibilizar uma opção para excluir os respectivos registros. Ao selecionar essa opção, uma janela modal deverá ser exibida solicitando a confirmação da operação de deleção. Se o usuário optar por não confirmar, a janela modal será simplesmente fechada e nenhuma ação será executada. Por outro lado, caso o usuário confirme a operação, deverá ser disparada uma requisição Ajax, utilizando o método POST, para o servidor, solicitando a exclusão do item correspondente.

#15 (NÃO resolvido pelo professor): Criar a **página de cadastro** onde os usuários poderão criar suas contas para usar a aplicação. Os dados necessários para o cadastro do usuário são: nome completo, email, senha, repetir senha, e curso (deverá ser mostrado um dropdown com os cursos cadastrados no sistema). Após um usuário se cadastrar, sua senha

deverá ser criptografada usando as diretrizes apresenta das nos slides.

#16 (NÃO resolvido pelo professor): Adicionar o jogo implementado no primeiro trabalho prático na página principal da aplicação, na rota / (raiz). Note que apenas usuários logados deverão ter acesso ao jogo. Quando um usuário estiver jogando o jogo e se deparar um com Game Over, sua aplicação deverá fazer uma requisição Ajax ao servidor para salvar os scores dos usuários na tabela game_sessions.

#17 (NÃO resolvido pelo professor): Criar uma página Ranking (rota /ranking) cotendo uma tabela com a pontuação dos 10 usuários distintos que mais pontuaram em seu jogo. A ordem dos usuários na tabela deve ser em ordem descrescente de acordo com a pontuação obtida por cada usuário.

#18 (Resolvido parcialmente pelo professor): Desenvolver um middleware que utiliza variáveis de sessão para impedir que usuários não autenticados acessem rotas restritas, ou seja, rotas que são permitidas apenas para usuários que estejam devidamente logados na aplicação. Caso o usuário não esteja autenticado, o middleware deverá bloquear o acesso e redirecioná-lo para a página de login.

#19 (Resolvido parcialmente pelo professor): Criar uma Navbar para a aplicação, usando o componente Navbar do Bootstrap. A lista de atalhos exibida na Navbar depende se o usuário está logado ou não na aplicação. Quando o usuário não está logado, são exibidos os atalhos **Sobre**, que direciona para a rota /about, **Login**, que leva para a página de autenticação, e **Criar Conta**, que leva para a página de cadastro. Quando o usuário está logado, os atalhos exibidos na Navbar são **Sobre**, que também direciona para a rota /about, **Jogar**, que leva para a página onde o usuário pode iniciar uma partida do jogo, e **Ranking**, que direciona para a página com a lista das melhores pontuações. Além disso, quando o usuário está logado, o canto direito da Navbar exibe seu nome, que, ao ser clicado, abre um menu suspenso (Dropdown do Bootstrap) contendo as opções **Efetuar logout**, **Alterar dados cadastrais** e **Alterar senha**. A opção **Alterar dados cadastrais** direciona para uma página onde o usuário pode atualizar seus dados pessoais, enquanto a opção **Alterar senha** leva para uma página que permite modificar a senha, a qual deve conter três campos: senha atual, nova senha e repetir nova senha. A alteração da senha só será permitida se o usuário informar corretamente a senha atual e se os campos de nova senha e repetir nova senha estiverem preenchidos com o mesmo valor.

Forma de Entrega

A entrega do trabalho deverá ser feita em duas etapas. A primeira etapa deverá constar as soluções dos exercícios 1 a 13, e deverá ser entregue até o dia o dia 27 de junho (sexta-feira). O código da primeira etapa deverá ser disponibilizado em uma pasta chamada **ExpTs** do repositório do Github. A segunda etapa deverá constar as soluções de todos os exercícios (incluindo todos os da primeria etapa) e deverá se entregue até o dia 11 de julho (sexta-feira). O código completo etapa deverá ser disponibilizado em uma pasta chamada **ExpTsFinal** do repositório do Github.

IMPORTANTE 1: As datas apresentadas não poderão ser adiadas, pois eu vou precisar de tempo hábil para corrigir os trabalhos e definir quem vai precisar fazer a prova final ou não.

IMPORTANTE 2: Será descontado alguns pontos caso o arquivo **.env** e os diretórios **node_modules** e **build** estejam no repositório do Github (isto é, não estejam listados no arquivo .gitignore). Por outro lado, também será descontado pontos caso o repositório não tenha um arquivo **.env.example**.

Para corrigir seu trabalho, eu vou seguir os seguintes passos: (ii) Fazer o **clone** de seu repositório do Github; (ii) Executar **npm install**; (iii) Copiar o seu arquivo **.env.example** do seu projeto para um arquivo **.env**; (iv) Editar as variáveis do .env, incluindo a variável DATABASE_URL, para que essa contenha os dados de acesso ao meu banco de dados local; (v) Rodar as migrações; (vi) Rodar a aplicação com **npm start**; (vii) Corrigir a aplicação; (viii) Procurar por outras implementações iguais para detectar casos de plágio.

IMPORTANTE 3: Serão descontados pontos caso eu não consiga rodar a aplicação a partir dos passos apresentados acima.

IMPORTANTE 4: Para detectar plágio, serão considerados principalmente os códigos que não eu não implementei durante as aulas. Alunos com códigos plagiados, em ao menos um dos exercícios acima, terão seu trabalho descartado e ficarão com nota final igual a ZERO (0.0) em todo o trabalho.