INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

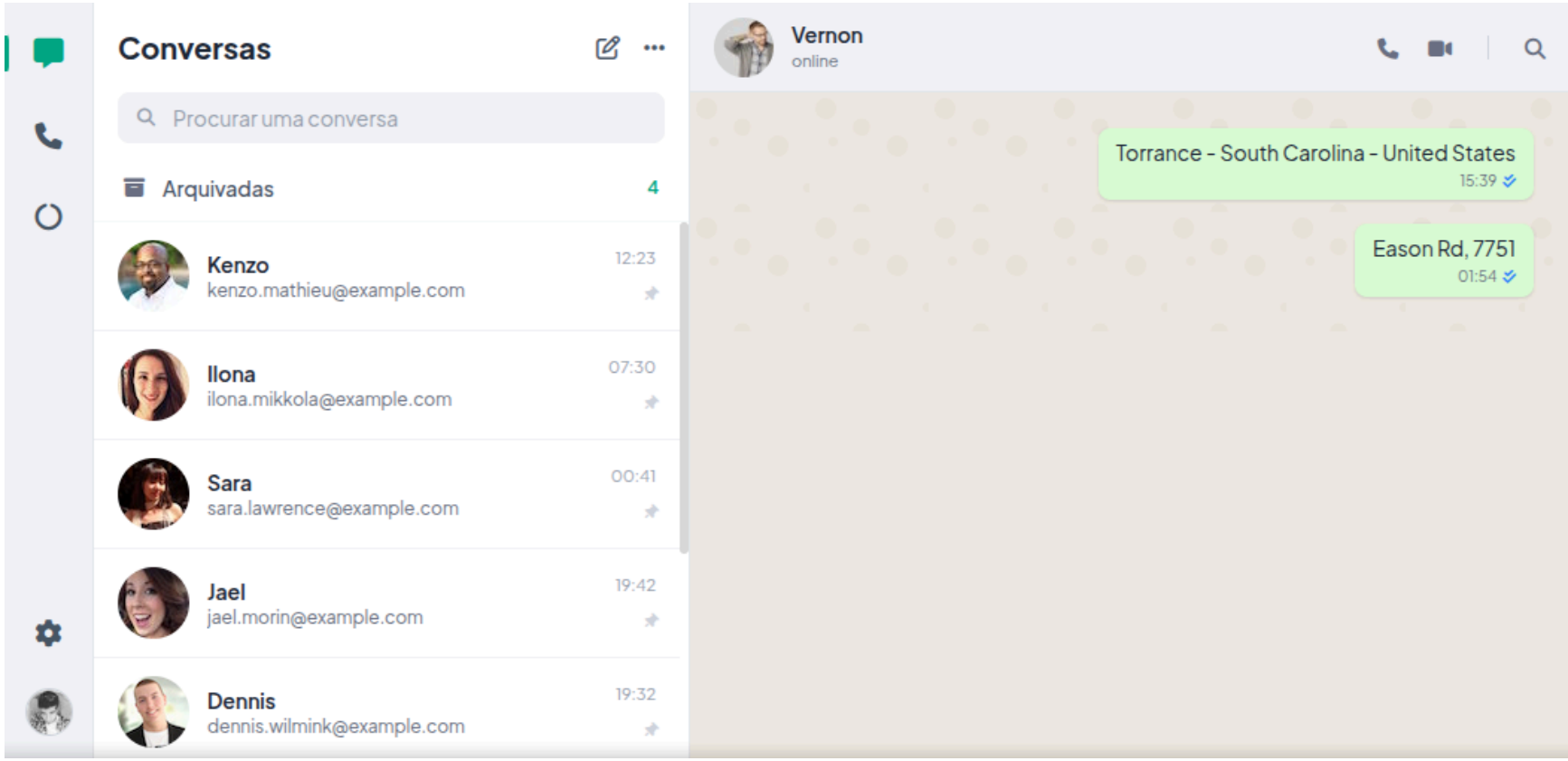
ENSINO MÉDIO INTEGRADO - INFORMÁTICA
Disciplina de Desenvolvimento Web
Avaliação A - 2º Bimestre

Gil Eduardo de Andrade

INSTRUÇÕES PARA RESOLUÇÃO DA AVALIAÇÃO

1. A prova é resolvida de forma individual ou em dupla, não havendo troca de informações entre os alunos, nem mesmo troca de material como cadernos, livros ou pen drive.
2. A interface da aplicação web deve conter, obrigatoriamente, os mesmos recursos e layout apresentados pelo professor, antes do início da avaliação, em sua codificação de resolução da prova.
3. Ao final da prova o aluno deve enviar o(s) link(s) do(s) repositório(s) para o e-mail do professor (gil.andrade@ifpr.edu.br)
4. O desenvolvimento das aplicações deve usar como base a aplicação Angular disponibilizada neste link https://github.com/Instituto-Federal-PR/vite-angular-tailwind (porta 12006) e a aplicação Svelte disponibilizada neste link: https://github.com/Instituto-Federal-PR/vite-svelte-tailwind (porta 12008).
5. A utilização do Docker é recomendada, visto que os repositórios disponibilizados já estão configurados para utilização do mesmo, sendo necessário apenas executar o comando: `docker compose up` para levantar as duas aplicações.
6. A estrutura de diretórios e arquivos dos dois projetos deve seguir o padrão apresentado a seguir, durante a descrição de cada uma das aplicações.
7. Todos os dados carregados, lista de usuários e lista de imagens devem ser obtidos através do acesso às APIs: https://randomuser.me/api (exemplo) e https://picsum.photos/v2 (exemplo), respectivamente. A primeira API será utilizada nas duas aplicações, e fornece nomes, fotos e dados pessoais. A segunda API será utilizada apenas na aplicação Svelte, onde existe um componente de Feed que apresenta um conjunto de imagens.

1º Aplicação: WhatsApp Desktop (Angular)

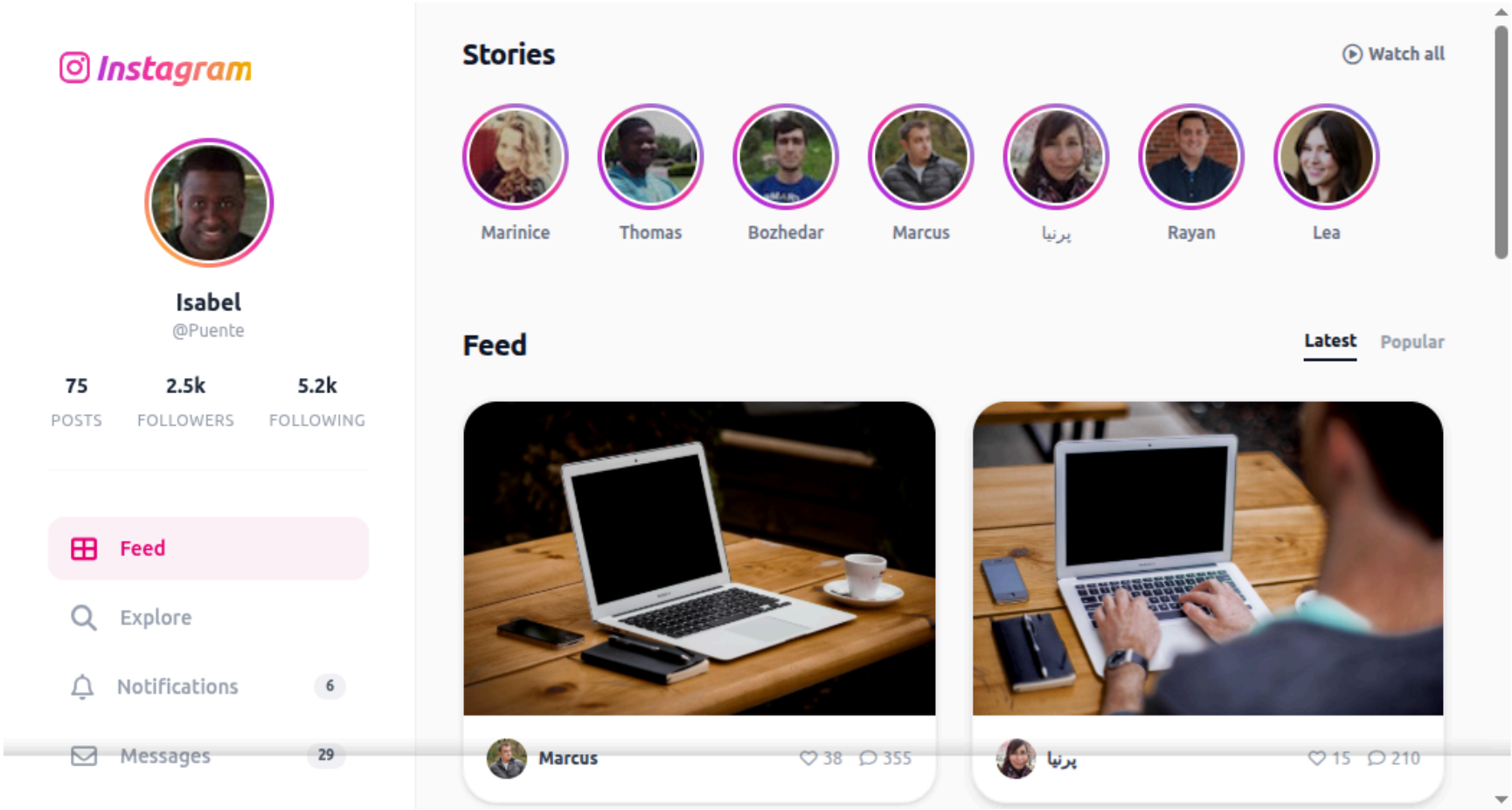


A aplicação Angular deve conter a seguinte estrutura de diretórios:

- **src/**
  - **api/**
    - api.urls.ts (Contém as URLs das APIs utilizadas)
    - user.api.ts (Responsável por efetuar a requisição HTTP a API de usuários aleatórios)
  - **components/**
    - chat-box.component.ts (Caixa de Conversas - esquerda da janela / todas as conversas)
    - chat-content.component.ts (Caixa com o conteúdo da conversa - direita da janela)
    - chat-item.component.ts (Itens que compõem o componente chat-box - foto e nome dos contatos)
    - header-content.component.ts (Caixa acima do conteúdo das conversas - foto e nome do contato/online)
    - side-menu.component.ts (Menu lateral esquerda - foto do usuário do whatsapp que está logado)
  - **models/**
    - user.model.ts (Define os tipos que representam os usuários e a lista de usuários)
  - **services/**
    - chat-selected.service.ts (Serviço compartilhado entre os componentes chat-item e header-content)
    - user.service.ts (Serviço que fornece a lista de usuários para os componentes)
  - **utils/**
    - random.ts (Utilitário que gera números aleatórios num intervalo especificado)

Para efetuar as requisições HTTP junto as APIs, no Angular, utilize, obrigatoriamente a biblioteca { HttpClient}, método get() - dentro do arquivo *api/user-api.ts*. Para as funcionalidades que exigem reatividade, ou seja, atualização do componente na tela, quando um valor é modificado, utilize, obrigatoriamente, o conceito de sinais (signal). Exemplos: carregamento da lista de usuário da API, que deve fazer com que o componente chat-box seja atualizado. Ou ainda, no serviço chat-selected, quando uma conversa é selecionada no componente chat-item, as informações nos componentes header-content e chat-content devem ser atualizados.

2º Aplicação: Instagram Desktop (Svelte)



A aplicação Svelte deve conter a seguinte estrutura de diretórios:

- **src/lib/**
  - **api/**
    - urls.ts (Contém as URLs das APIs utilizadas)
    - phto.service.ts (Responsável por efetuar a requisição HTTP a API de fotos aleatórias)
    - user.service.ts (Responsável por efetuar a requisição HTTP a API de usuários aleatórios)
  - **components/**
    - Feed.svelte (Caixa de Feeds - direita da janela / contém a lista de fotos dos outros usuários)
    - FeedItem.svelte (Itens que compõem o componente Feed - foto, foto do usuário, curtidas, etc)
    - Menu.svelte (Menu lateral esquerda abaixo - contém opções Feed, explore, etc)
    - Profile.svelte (Caixa lateral esquerda acima - contém foto do usuário logado, nome, seguidores, etc)
    - Stories. svelte (Caixa superior direita - contém as fotos dos usuários que possuem storie)
    - StoriesItem.svelte (Itens que compõem o componente Stories - foto e nome)
  - **models/**
    - photo.model.ts (Define o tipo que representa as fotos carregadas da API)
    - user.model.ts (Define os tipos que representam os usuários e a lista de usuários)
  - **states/**
    - photo-state.svelte.ts (State que gerencia a lista de fotos fornecidas aos componentes)
    - user-state.svelte.ts (State que gerencia a lista de usuários fornecida aos componentes)
  - **utils/**
    - random.ts (Utilitário que gera números aleatórios num intervalo especificado)

Para efetuar as requisições HTTP junto as APIs, no Svelte, utilize, obrigatoriamente a função fetch() do JavaScript dentro dos arquivos *api/photo-service.ts* e *api/user-service.ts*. Para as funcionalidades que exigem reatividade, ou seja, atualização do componente na tela, quando um valor é modificado, utilize, obrigatoriamente, o conceito de states ($state()). Exemplos: carregamento da lista de usuário da API e lista de fotos da API, que deve fazer com que os componentes *"Stories"* e *"Feed"* sejam atualizados.