UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Aplicação Turistando

Relatório Final

Integrantes:

- Bruno Oliveira Sinhoroto
- Davi Augusto Silva
- Gustavo Vinícius Alba
- Marcelo Junio de Oliveira Teixeira
- Otávio Almeida Leite
- Paulo Kiyoshi Oyama Filho

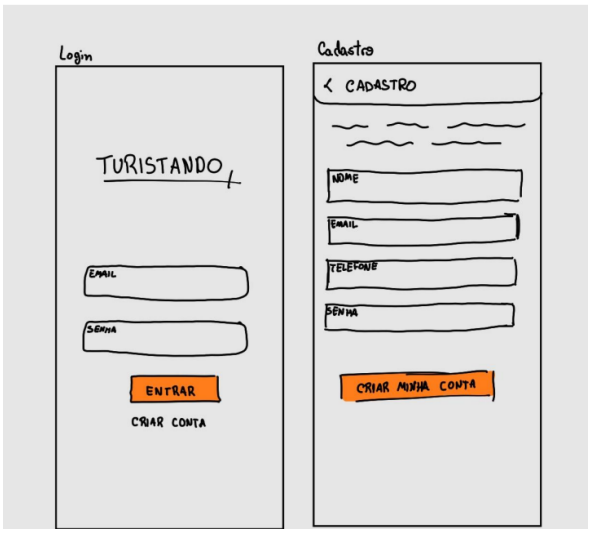
Concepção do Sistema

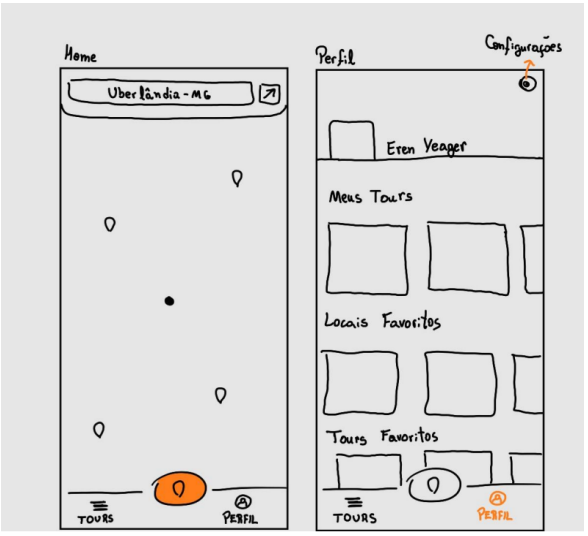
Descrição da Aplicação

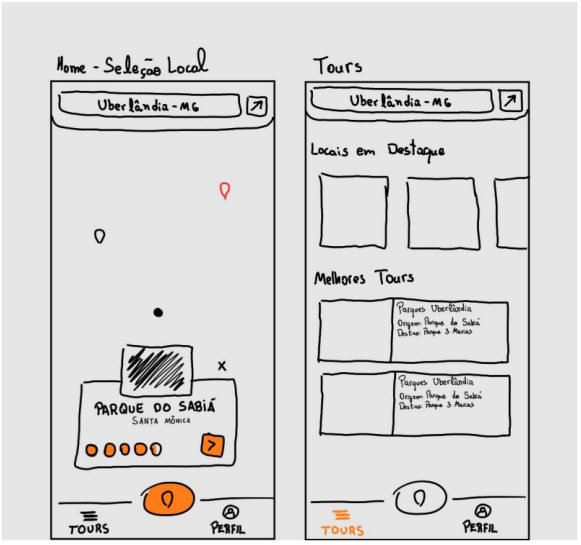
O Turistando é um aplicativo de visualização de pontos turísticos e planejamento de viagens. Por meio dele, os usuários podem ver em um mapa os pontos turísticos e sua localização para montar um tour a se fazer por eles. Nisso, os tours criados por usuário se tornam públicos para que qualquer outros usuários da plataforma possam visualizá-los.

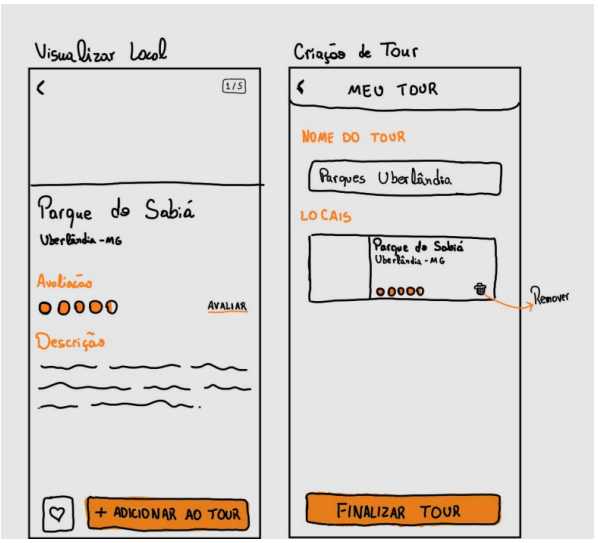
Os fluxos de utilização são basicamente de controle de contas de usuários e configurações, visualização de listagem de localidades, visualização de localidades em mapas interativos, criação e visualização dos tours.

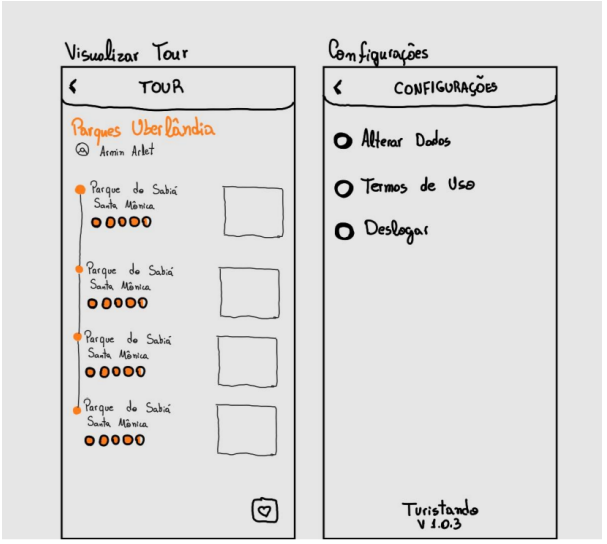
A partir da ideia inicial, foram feitos os seguintes rascunhos do protótipo de interface, a fim de ter a percepção do usuário final que irá utilizar a aplicação.


Login
TURISTANDO
EMAIL
SENHA
ENTRAR
CRIAR CONTA
Cadastro
CADASTRO
NOME
EMAIL
TELEFONE
SENHA
CRIAR MINHA CONTA

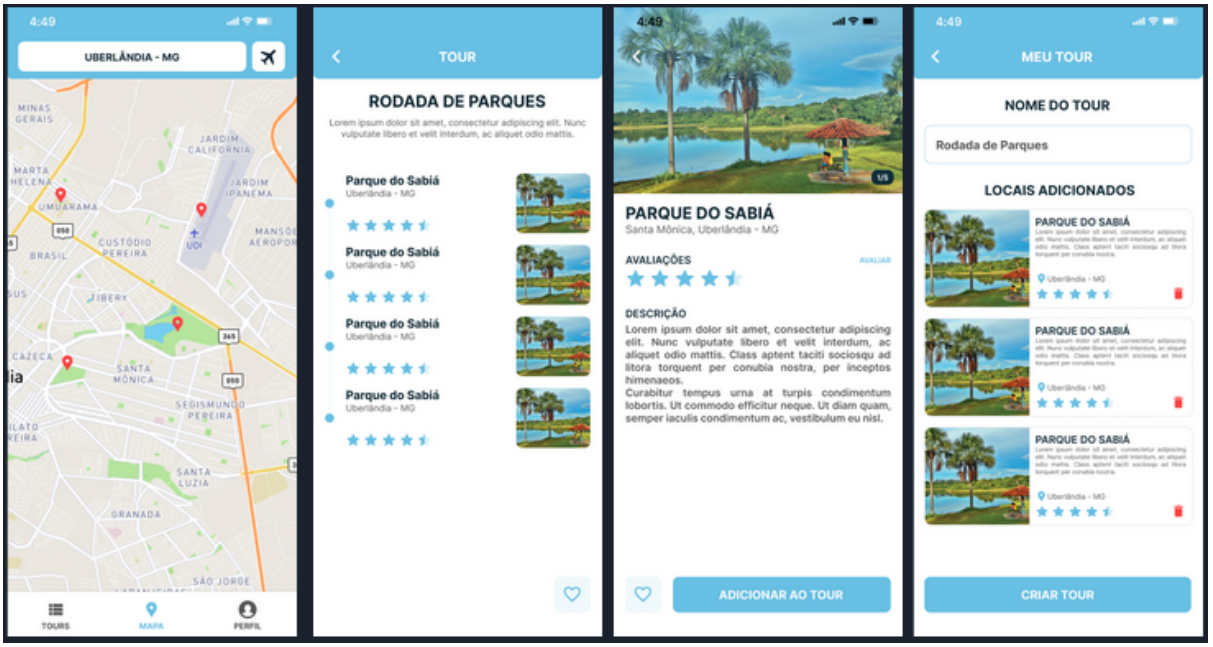

Home
Uberlândia - MG
TOURS
PERFIL
Perfil
Configurações
Eren Yeager
Meus Tours
Locais Favoritos
Tours Favoritos
TOURS
PERFIL


Home - Seleção Local
Uberlândia - MG
PARQUE DO SABIÁ
SANTA MÔNICA
TOURS
PERFIL
Tours
Uberlândia - MG
Locais em Destaque
Melhores Tours
TOURS
PERFIL


Visualizar Local
1/5
Parque do Sabiá
Uberlândia - MG
Avaliação
AVALIAR
Descrição
+ ADICIONAR AO TOUR
Criação de Tour
MEU TOUR
NOME DO TOUR
Parques Uberlândia
LOCAIS
Parque do Sabiá
Uberlândia - MG
Remover
FINALIZAR TOUR


Visualizar Tour
TOUR
Parques Uberlândia
Armin Arlet
Parque do Sabiá
Santa Mônica
Parque do Sabiá
Santa Mônica
Parque do Sabiá
Santa Mônica
Parque do Sabiá
Santa Mônica
Configurações
CONFIGURAÇÕES
Alterar Dados
Termos de Uso
Deslogar
Turistando
V 1.0.3

Após isso, houve o refinamento do conceito elaborado, resultando na criação de um projeto no Figma, que pode ser acessado pelo seguinte link, uma parte do figma está apresentado abaixo.



Diferencial Competitivo

Como principal diferencial o Turistando possui a personalização de tours turísticos que podem ser feitos por qualquer um, podendo assim incluir viajantes experientes ou nativos da região, dispensando um guia turístico uma vez que a rota já estará pronta através do GPS, eliminado o custo do guia e de um planejamento turístico.

O Turistando também possuirá um sistema de gamificação que beneficiará aqueles que criarem os melhores tours dentro do app, incentivando assim tours mais bem elaborados ou desejados pela comunidade.

Definição do Ferramental

São as principais ferramentas a serem utilizadas, atribuídas às respectivas finalidades:

- Front-end mobile: Linguagem Dart com Framework Flutter
- Back-end: Linguagem Python com Framework *FastAPI*
- Banco de Dados: PostgreSQL
- Versionamento: Git com GitHub
- Organização e Planejamento do Projeto: Notion
- Prototipação de Interface: Figma
- Criação de Diagramas: Diagrams.net, dbdiagram.io

Requisitos

Requisitos Funcionais

De usuário

1. **Acesso do Cliente**
   O Cliente acessa o aplicativo através da senha e do email já cadastrados, caso não houver cadastrado então deverá fazer o cadastro[2].
2. **Acesso Inicial - Cadastro do cliente**
   O cliente deverá fornecer seu nome, email, telefone e uma senha para realizar o cadastro no aplicativo.
3. **Módulo Home - Inicial**
   O aplicativo mostrará o usuário em uma mapa possibilitando sua visualização, assim como a visualização de pontos turísticos perto dele, além disso é possível ver outros tours criados por outros usuários e melhores pontos turísticos.
4. **Módulo Home - Interações**
   O usuário poderá se mover no mapa clicando em pontos turísticos, e poderá ver tours feitos por outros usuários.
5. **Módulo Perfil**
   O usuário pode ver os tours feitos anteriormente, além disso ele pode ver os tours criados por si mesmo nesta página, seus locais favoritos e suas configurações.
6. **Página Ponto Turístico - Mais detalhes**
   O usuário poderá se mover no mapa clicando em pontos turísticos, e poderá ver tours feitos por outros usuários.
7. **Módulo Tours**
   O usuário poderá ver os locais em destaque e os tours também em destaque.
8. **Criação de Tours**

O usuário pode criar seu próprio tour após clicar em em “Adicionar ao Tour” na páginas de detalhes do local.

9. **Visualização de Tour**
   O usuário pode ver o tour que será iniciado, como os locais que serão visitados e o título do tour.
10. **Módulo Configuração**
    O usuário pode ver os seus direitos no app e seu comportamento dentro dele por meio dos termos de uso, ele pode deslogar da conta que estará no momento e poderá alterar os dados, caso ele erre algo dos seus dados cadastrais.

De sistema

1. **Acesso do Cliente**

    1. Para logar no app o cliente **deve** estar cadastrado.
        i. Caso **não** esteja, **vá** para o requisito [2].
    2. Para isso, o cliente **deve** informar seu email cadastrado.
    3. O cliente **deve** informar sua senha cadastrada .

2. **Acesso Inicial - Cadastro do cliente**

    1. O cliente **deve** fornecer seu nome.
    2. O cliente **deve** fornecer seu email.
    3. O cliente **deve** fornecer seu telefone.
        i. Seu telefone **deve** ser colocado no padrão, **(xx) xxxxx-xxxx**.
    4. O cliente **deve** fornecer sua senha.
        i. Sua senha **deve** ter no mínimo 8 dígitos.

3. **Módulo Home - Inicial**

    1. Por **DEFAULT**, teremos a cidade do cliente como padrão no primeiro acesso.
        i. Além disso, ela **irá** ter uma barra de navegação no canto inferior
        ii. E também um cabeçalho na parte superior contendo a cidade em questão e o atalho para a criação de um tour.
    2. O módulo home **deve** fornecer um mapa da região expressa no cabeçalho.
    3. A página **deve** mostrar mapa com a localização atual do usuário.
    4. A página **deve** mostrar os pontos turísticos no mapa[3.2].
    5. Caso seja clicado no ícone “Perfil” na barra de navegação, o app **terá** de navegar para o módulo Perfil[4].
    6. Caso seja clicado no ícone “Tours” na barra de navegação, o app **terá** de navegar para o módulo Tours[7].

7. Caso o cliente deseje alterar sua cidade **deve** clicar no cabeçalho.
8. Caso o cliente deseje criar um tour **deve** clicar no ícone de avião no cabeçalho.

4. Módulo Home - Interações

1. Ao clicar em algum ponto turístico[3.3].
2. **Deverá** ser criado um card no canto inferior
3. Este card **deve** ser composto por uma imagem do ponto turístico, seu nome, sua localização, sua avaliação e um botão para mais detalhes.
4. Ao clicar na cidade no cabeçalho, será possível trocar de cidade, podendo o usuário escolher sua localização atual ou escolher no mapa.

5. Módulo Perfil

1. Ao entrar neste módulo, **deverá** ser mostrado a foto de perfil assim como seu nome na mesma linha e um botão de configuração.
    i. Ao clicar no botão de configuração, o usuário **deverá** ser encaminhado para o módulo de configuração[10].
    ii. Caso não haja uma foto de perfil do usuário, **deverá** ser mostrado uma imagem default previamente escolhida.
2. Além disso, **deve** mostrar ter 3 componentes, sendo elas;
    i. Meus Tours.
        1. **Deverá** ter uma listagem representando os tours do usuário.
    ii. Locais Favoritos.
        1. **Deverá** ter uma listagem representando os locais favoritos do usuário.
    iii. Tours Favoritados.
        1. **Deverá** ter uma listagem representando os tours favoritos do usuário.
3. A página pode ser scrollável, **caso seja necessário** para determinadas telas.
4. A barra de navegação[3.0.1] **deve** permanecer nesta tela.

6. Página Ponto Turístico - Mais detalhes

1. Ao carregar a página de detalhes ela **deve** ser composta por:
2. Uma ou mais imagens do ponto turístico, **podendo** ser mostrado por um *carousel*.
3. Um título e sua localização.
4. Um componente para visualizar sua avaliação, assim como um botão para que o próprio usuário possa avaliar o lugar.
5. Uma breve descrição do lugar.

6. No canto inferior, deve ter um ícone para dar “like” (curtir) o ponto turístico, e outro botão que **possibilitará** adicionar este ponto no tour feito pelo próprio usuário.
7. Se for clicado para “Adicionar ao tour”, o ponto turístico será adicionado à lista de criação de tour.

7. Módulo Tours

1. O cabeçalho[3.0.2] **deverá** permanecer nessa tela tendo as mesmas funcionalidades e a barra de navegação[3.0.1] também **deve** permanecer.
2. Esta página **deve** mostrar 2 partes principais:
3. A primeira será a de “Locais”, onde serão mostrados os locais com melhor avaliação e os locais em destaque.
4. A segunda parte será a de “Tours”, onde terá uma lista de tours em que a ordem de listagem será de acordo com as avaliações dadas pelos usuários.

8. Criação de Tours

1. É iniciado **após** clicar no ícone de avião no cabeçalho da página principal e da página de tour.
2. Para criação de um tour **deve** ser dado um título, esse título deve ser editável.
3. Ao clicar no botão “Adicionar tour”na tela de Detalhes de um Ponto Turístico, o ponto turístico irá aparecer em uma listagem
4. Os locais devem ser exibidos em uma listagem, cada ponto turístico nessa coluna **deve** ser passível de **deletação**
5. Cada ponto turístico **será** mostrado como um Card contendo uma imagem, um título, uma avaliação e a sua localização.
6. Ao final da página **deve** haver o botão “Finalizar Tour”, que indicará a finalização da criação do Tour e após isso o tour poderá ser usado pelo usuário e por outros usuários da plataforma.
7. Ou seja, o tour **deverá** ser disponibilizado para outros usuários da plataforma.

9. Visualização de Tour

1. Essa página **deverá** mostrar o título do Tour.
2. Ele também **deverá** mostrar o nome do criador do Tour.
3. Além disso, ele **deve** mostrar os pontos turísticos que fazem parte do tour.
    i. Esses pontos **devem** mostrar uma imagem que representa ele, sua avaliação, sua localização e um título.
4. Cada usuário **poderá** curtir o Tour, para que fique guardado caso ele goste e ajude para futuras avaliações.

10. Módulo Configuração

1. Este módulo **deve** mostrar as configurações do app,
2. **Deve** ter a opção de alterar dados.
3. Os dados passíveis de serem alterados **serão** o nome, a senha, o email e o número de telefone.
4. **Deve** dar acesso aos Termos de Uso do aplicativo.
5. **Deve** ter a opção de deslogar do app.
6. **Deve** apresentar a versão corrente do aplicativo.

Requisitos Não Funcionais

De produto

1. O aplicativo turistando **deve** rodar somente em celulares **Android ou IOS**, excluindo os demais.
2. O Aplicativo **deve** rodar em uma quantidade de 60 quadros por segundo ou **60 fps**.
3. Em casos que necessitam de acesso à internet no aplicativo, como carregamento de dados e imagens, **o tempo máximo de carregamento deve ser 30 segundos**.
4. Em casos de falta de internet ou instabilidade de conexão, o aplicativo **deve** informar o usuário dessa condição atual.
5. O carregamento do mapa no módulo Home **deve** ser somente no **escopo de cidade**.
6. O aplicativo **deve** deixar claro onde os dados do usuário serão usados nos Termos De Uso.
7. O sistema tem como público alvo pessoas que estejam viajando e/ou planejando uma viagem, possibilitando um roteiro de viagem.
8. O app **deve** ocupar **menos que 50 MB** de memória para uso.

Externos

1. Por vias legais é **expressamente proibido**, qualquer tipo de palavra ou nome com conotação sexual e ofensivo para outros.
2. Além disso, é também é **expressamente proibido** o uso de imagens obscenas ou com caráter pejorativo/ofensivo.
3. Estes requisitos acima [1-2] são motivos **direto** de exclusão da conta no aplicativo.

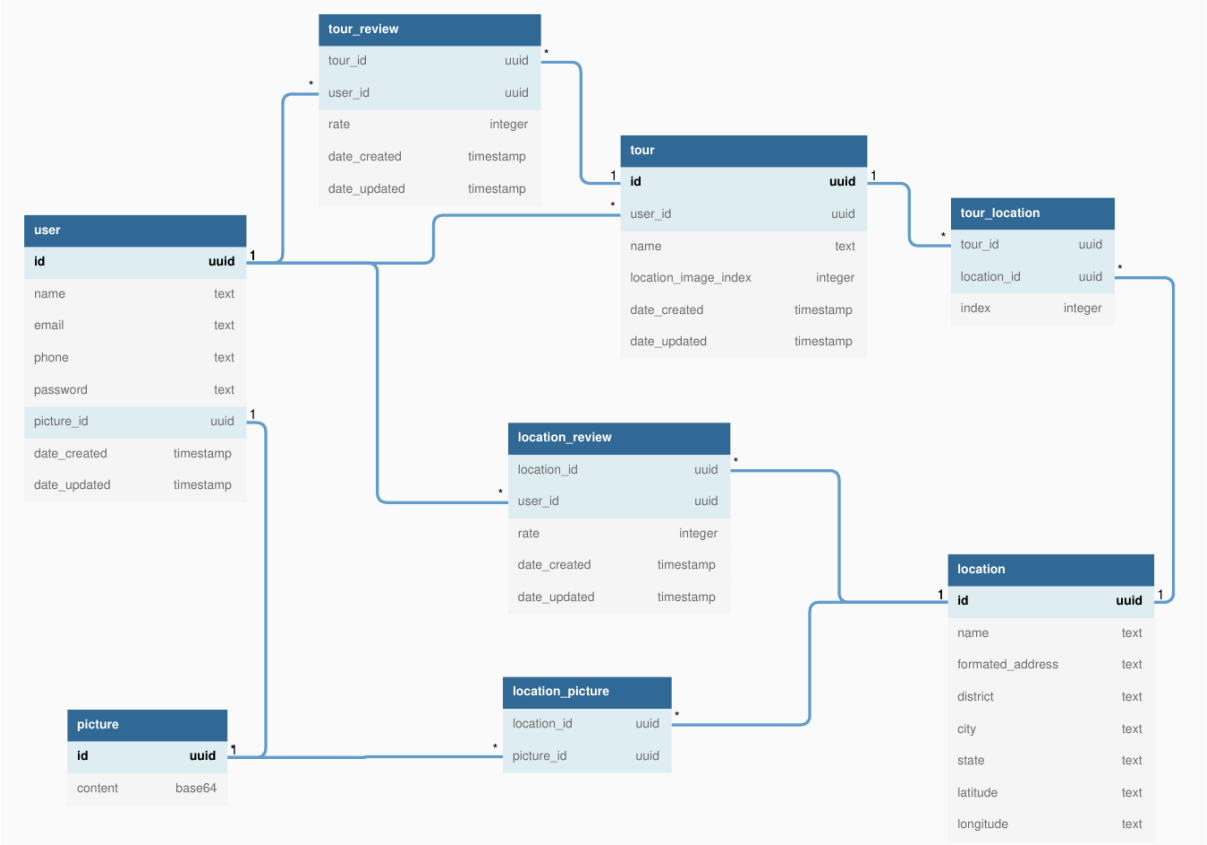
Projeto Arquitetural

Como se trata de um sistema online, a arquitetura do projeto é definida em dois pilares principais: o software que roda no cliente, responsável pelo controle de estado e interfaces (*front-end*) e o software que roda no servidor, responsável pela persistência e controle dos dados num banco de dados, assim como pela implementação das regras de negócio e modelagem do domínio em código (*back-end*). O *Front-end* interage com o *back-end* através de uma API HTTP, e os pontos de interação são chamados de *endpoints*.

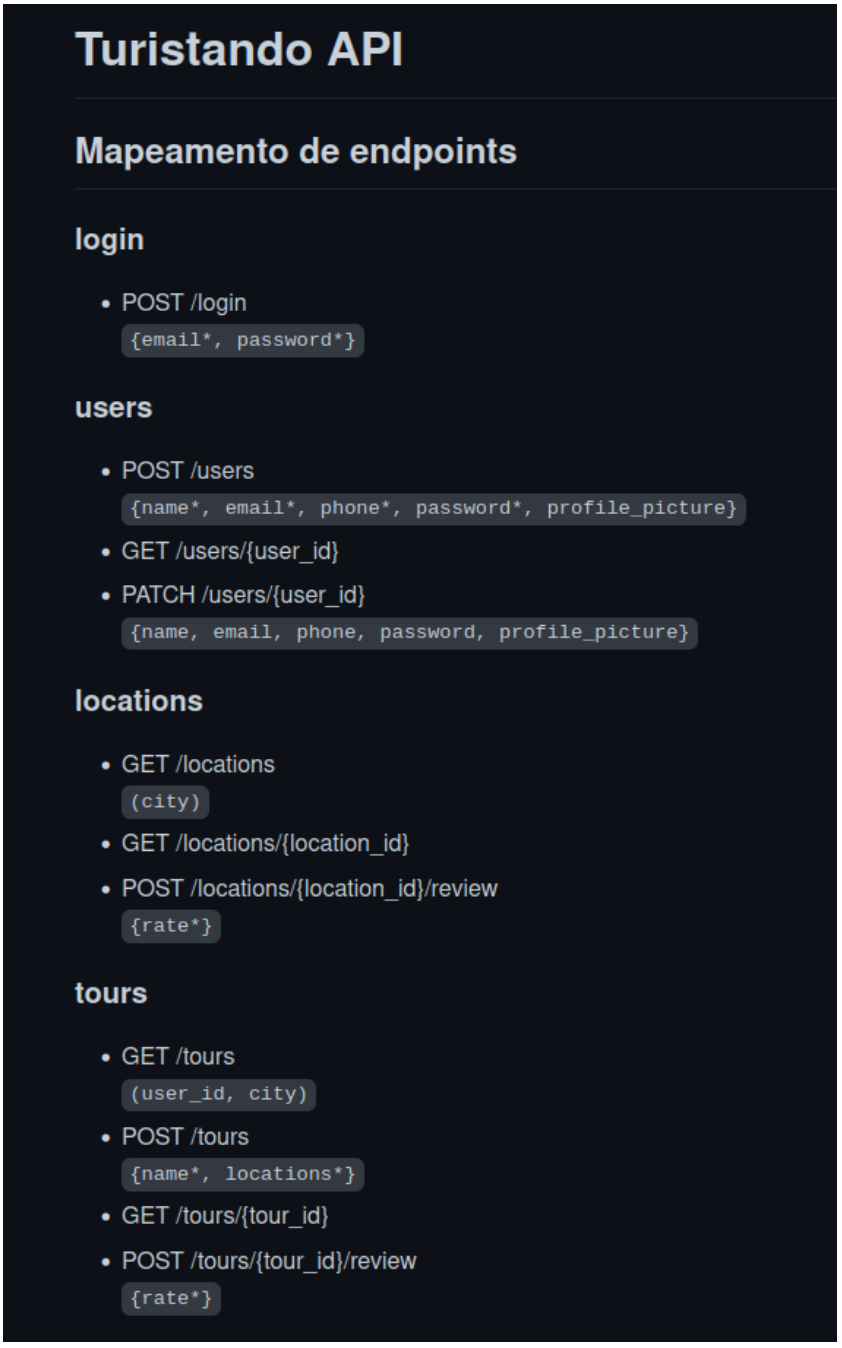
Com relação aos diagramas, foi necessário uma reunião com os responsáveis pelo back e front para se definir como seria a arquitetura, esses decidiram fazer de forma modularizada através de subpacotes como o “localidade”.

Assim sendo, foi-se mapeado cada subpacote do código para o diagrama de pacotes, quanto ao diagrama de implantação foi-se pensando na forma menos custosa mas ainda benéfica para o sistema.

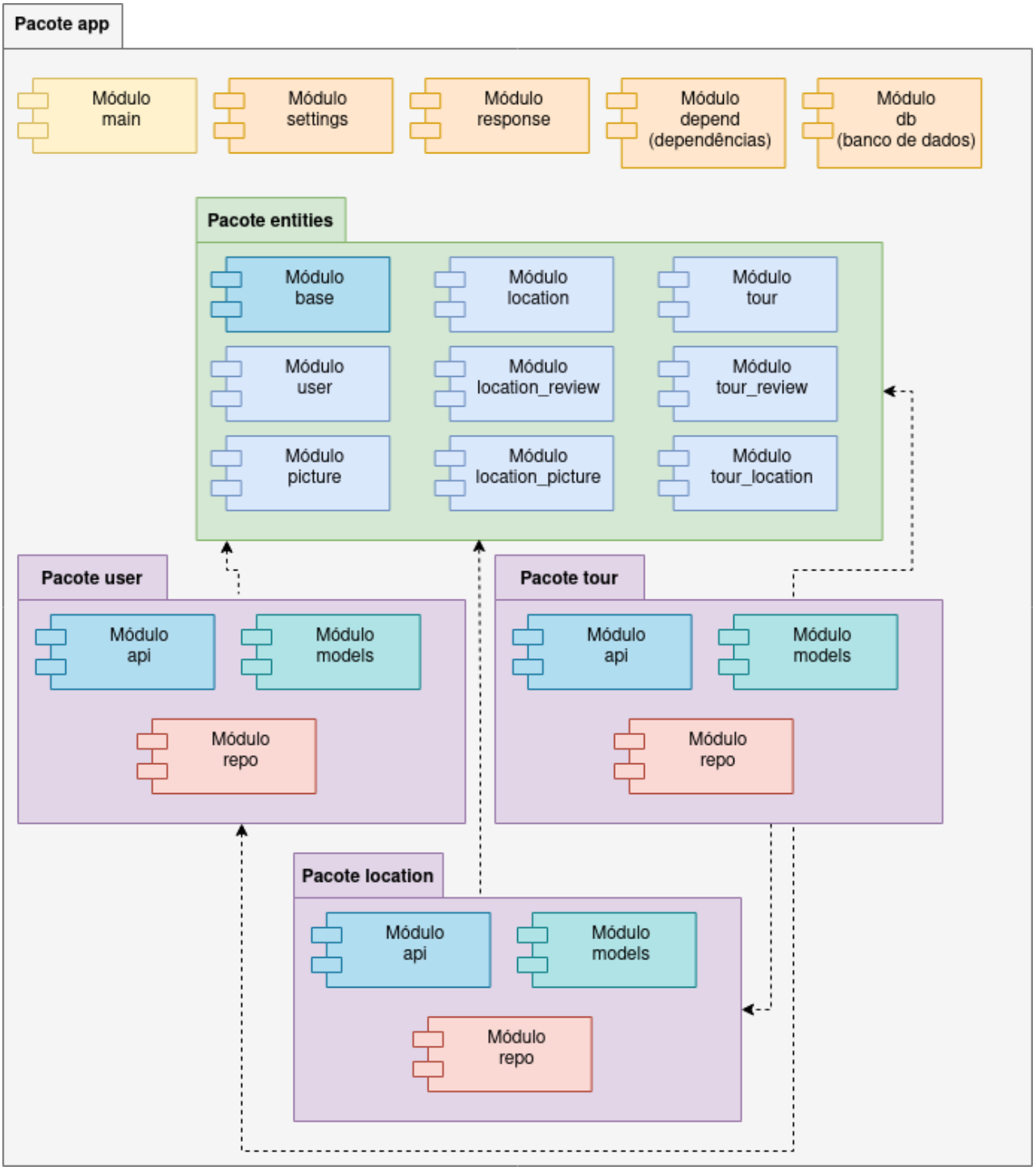
O começo da modelagem do projeto estrutural foi pelo mapeamento do banco de dados. Estruturando as entidades necessárias do banco relacional de acordo com os requisitos:



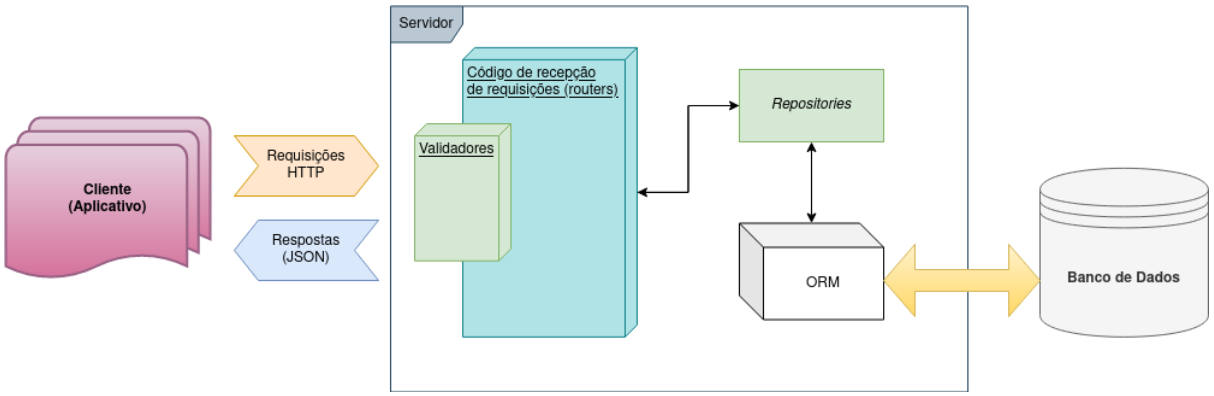
Em seguida, evoluindo em direção à estrutura da API propriamente dita, os requisitos foram revisitados e foram estruturados os endpoints envolvidos em cada subdomínio, sendo um subdomínio um conjunto de estruturas conceitualmente mais próximas dentro do domínio da aplicação. Os endpoints levantados foram:



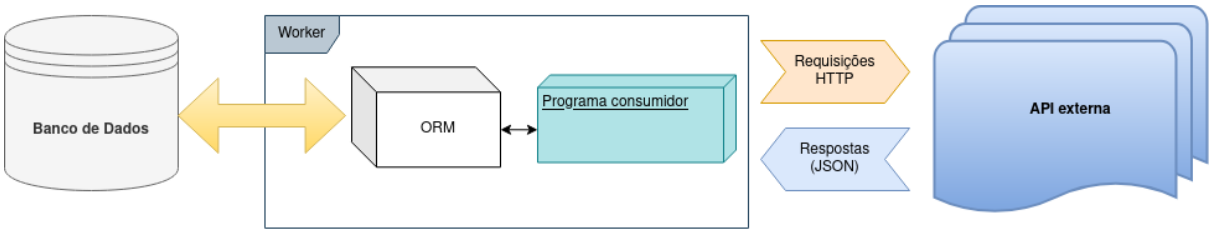
A partir dos endpoints levantados, foi iniciado o desenvolvimento do código. Estruturando e tomando as primeiras decisões específicas do projeto, foi possível traçar uma perspectiva mais concreta sobre um diagrama de pacotes do sistema do *backend*. Eis o diagrama de pacotes:



A partir da estruturação interna do sistema do *backend* é possível também projetar as interações externas a esse sistema, com os outros. Em primeiro lugar, as interações cliente, servidor, banco de dados:



A interação do sistema com o *worker*, um programa a parte que povoará o banco de dados com dados consumidos de uma fonte externa:

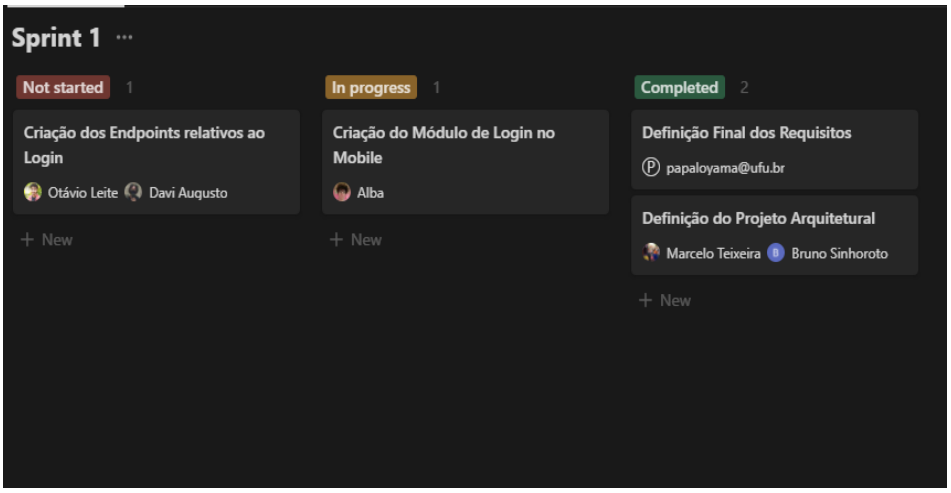


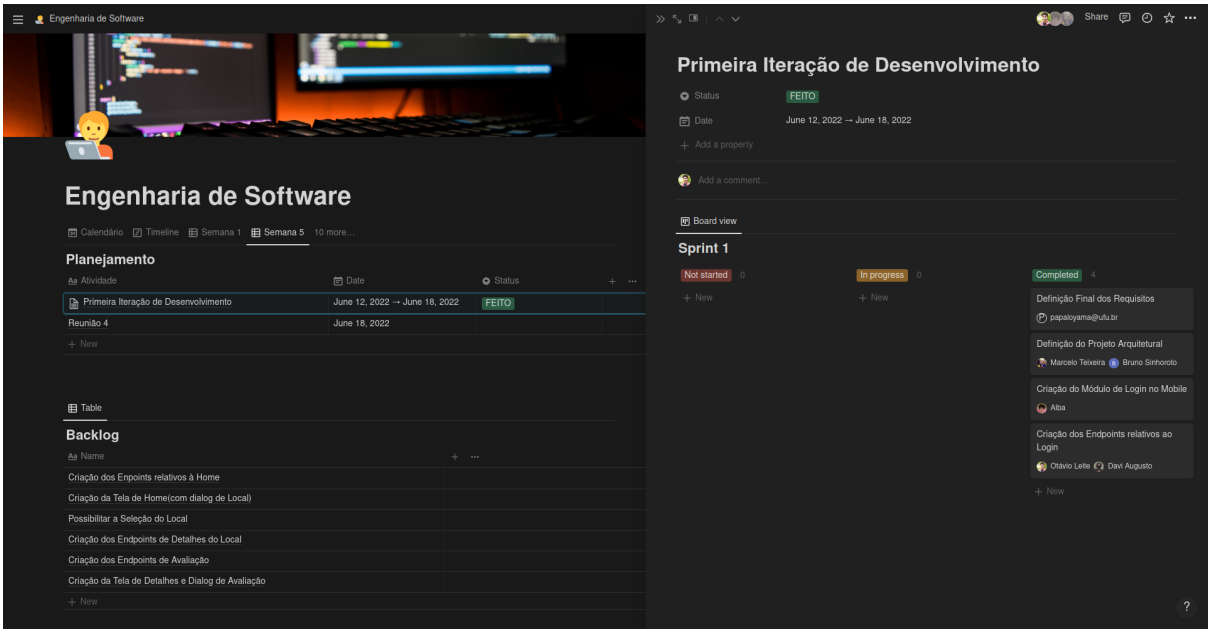
Um planejamento de implantação simples envolveria o seguinte: como se trata de um aplicativo mobile se fez necessário o uso de um dispositivo android e ios para os testes, um computador para a gerência, dois para atendimentos, uma impressora para relatórios e afins e um modem para distribuir internet e rede wifi para os dispositivos mobile acessarem o app.

Planejamento de Gestão

A ideia do planejamento era ser organizado por meio do aplicativo Notion. Inicialmente foram feitas atividades separadas relacionadas ao levantamento de requisitos, concepção do trabalho, criação de protótipos de interface, definição do ferramental e projeto arquitetural.

Após isso, começaram as iterações de planejamento, com sprints de 1 semana, nas quais teriam pequenas implementações e uma reunião semanal toda quinta-feira. A cada iteração era criado um novo board com as tarefas a serem feitas.





Assim como na metodologia Scrum, fazia-se presente o uso das sprints, dessa forma cada um possuía um backlog do produto de tarefas para serem cumpridas.

Mesmo definido os prazos das sprints e entregas durante essas cerimônias, o planejamento sofreu por várias adaptações devido às dificuldades de conciliação entre as rotinas de cada membro do grupo. Além disso, foi difícil reunir todos os integrantes em um mesmo dia, muitas das vezes as reuniões foram realizadas com um integrante a menos.

Ao seguir uma abordagem ágil, foi feita a análise dos requisitos mais importantes para que fossem implementados primeiro. Para cada requisito quebramos em componentes menores, diminuindo a sua granularidade e podendo assim dividi-los nas sprints.

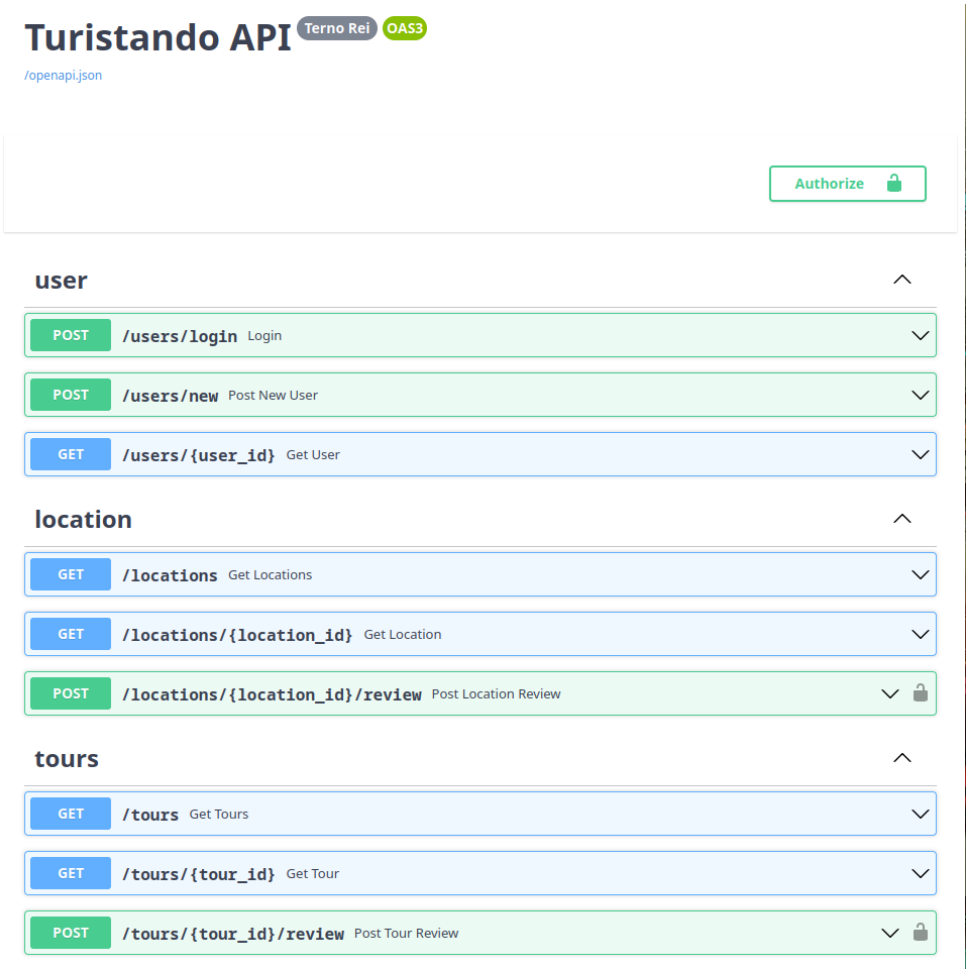
Além disso, buscamos estabelecer prazos para com esses requisitos.

Com relação a estrutura do Scrum, não houve um Scrum Master e Product Owner, contudo a cada semana algumas pessoas tomaram “a dianteira” e se encarregaram dos requisitos de todos, promovendo o planejamento e a sua boa execução.

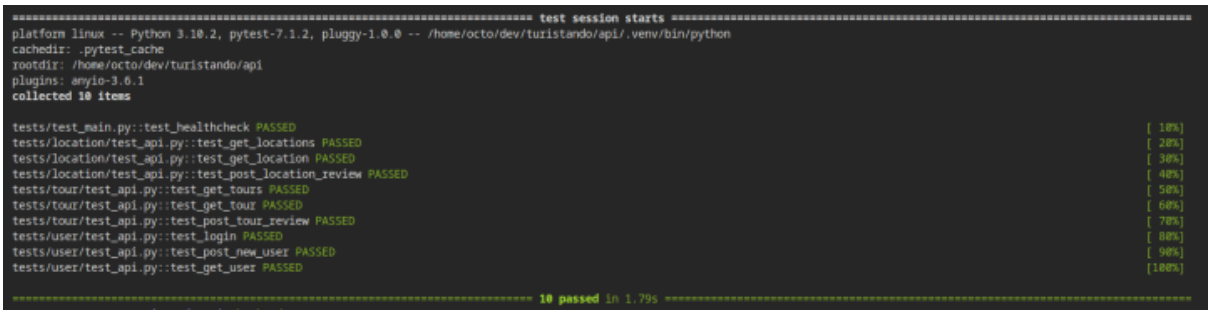
Na prática, houveram reuniões semanais para alinhar o andamento do projeto. As Iterações de Desenvolvimento começaram a ser realizadas a partir de 12 de junho. O plano levava em conta tempo de dedicação durante todo um dia, ainda que não diariamente. Houve a organização de um board kanban para cada iteração de desenvolvimento.

Resultado das Implementações

No campo do back-end, foram criados alguns endpoints para prover as informações para a aplicação cliente, podendo ser visualizado por meio do padrão OpenAPI com a interface do Swagger.



Juntamente, foram criados testes unitários para a validação do funcionamento das rotas, por meio da biblioteca de testes *pytest*. Os testes validam os diferentes caminhos possíveis para cada rota da aplicação, ou seja, verificam o retorno para casos de sucesso e casos de erro.



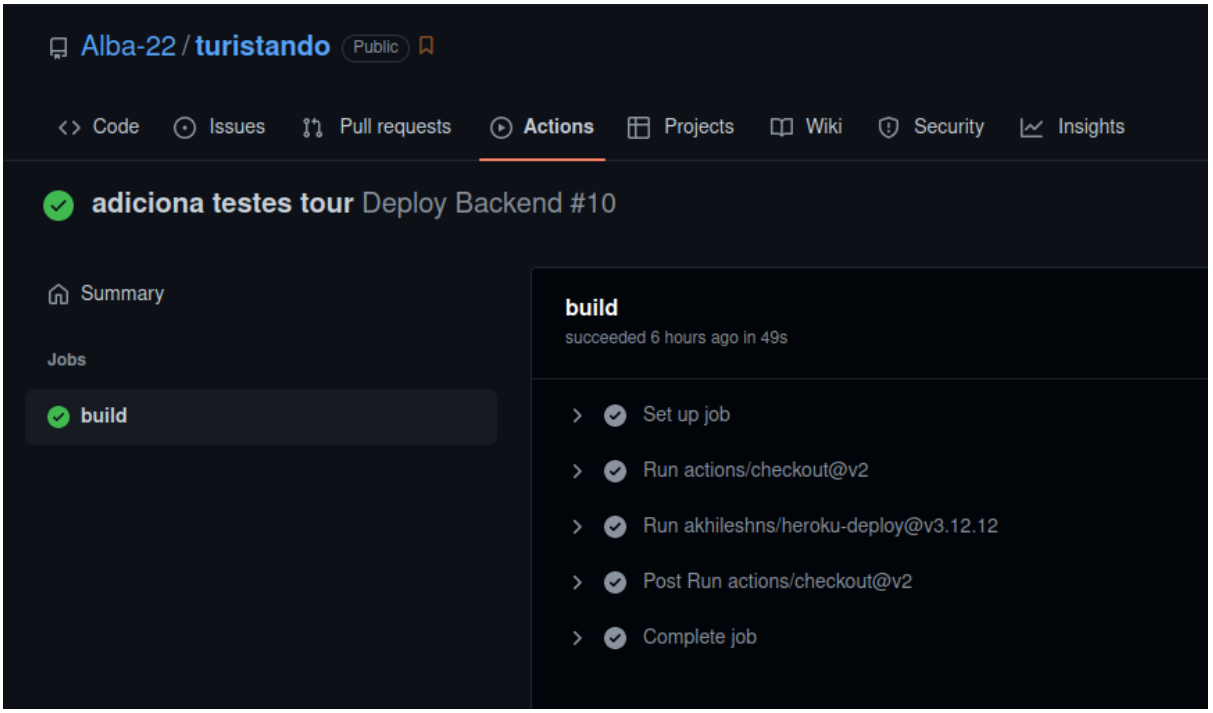
Estrutura geral de um teste de endpoint:

```python
def test_login(test_client: TestClient, user_in_db: User):
    """
    Testa rota de login.

    Caso 1: 200 - login com sucesso.
    Caso 2: 404 - usuário inválido.
    Caso 2: 404 - senha inválida.
    """

    test_cases: list[dict] = [
        {
            "input": {
                "username": user_in_db.email,
                "password": "Abcd1234#",
            },
            "expected": (200, {"token_type": "bearer"}),
        },
        {
            "input": {"username": "invalid_email@teste.io", "password": "1234"},
            "expected": (
                403,
                {"error_code": 1, "message": "Email ou senha inválidos."},
            ),
        },
        {
            "input": {
                "username": user_in_db.email,
                "password": "wrong_password",
            },
            "expected": (
                403,
                {"error_code": 1, "message": "Email ou senha inválidos."},
            ),
        },
    ]
    for test_case in test_cases:
        response = test_client.post("/users/login", data=test_case["input"])
        response_json = response.json()
        expected_status_code, expected_json = test_case["expected"]
        assert response.status_code == expected_status_code
        for key, value in expected_json.items():
            assert response_json[key] == value
```

Ademais, para o deploy do backend, foi feito uso de integração contínua por meio do GitHub Actions, o que possibilitou o lançamento simples e rápido de novas versões do sistema.



No campo da aplicação mobile, o *front-end* do sistema, foram criadas as telas de login, cadastro, tela inicial com mapa, tela de detalhes de um local, listagem de tours e criação de tours. No momento, nenhuma das telas no aplicativo está fazendo comunicação com o servidor, estando todas com dados provisórios para demonstração.

A exibição do aplicativo funcionando será melhor demonstrada na apresentação.

Desafios encontrados

Um dos principais desafios encontrados foi o mapeamento da arquitetura para os diagramas necessários, uma vez que o próprio projeto com seus pacotes ainda não estavam definidos, impossibilitando o mapeamento direto. Dessa forma, como até mesmo os pacotes estavam sendo feitos por diferentes pessoas, se fez necessário uma discussão para se especular como seria a arquitetura, porém viu-se durante a prática que a maneira escolhida sofreria várias modificações tendo assim que se mudar os diagramas em questão.

Além disso, tivemos dificuldade de comunicação entre os integrantes da equipe, tanto para reunião (sprint) quanto para outras atividades mais

especializadas e dificuldade em estimar o prazo com precisão, pois muita das vezes chegamos ao prazo com metade do produto.

E ainda, a produção de documentos detalhados, tivemos a dificuldade de integrar os documentos das partes da entrega , como cada um escrevia a parte que fazia, muita das vezes a formatação era diferente, então havia a necessidade de reformatação e integração para um formato padrão.

Outro grande desafio encontrado foi a organização das reuniões, uma vez que as rotinas do grupo não batiam entre si , o que atrapalhava as reuniões das sprints exigindo alguns sacrifícios e adaptações das sprints definidas durante a semana.

Analisando o processo que passamos podemos concluir que a falta de um papel intermediário que liga, por exemplo, o back end com o front, como o Scrum Master, se faz necessário, então uma figura mesmo que com rotatividade semanal seria muito necessário.

Além disso, vimos com as sprints que as entregas, os artefatos produzidos, não eram muito claros ou nem muito significativos. Portanto, uma parte de melhoria seria reuniões diárias, haja vista que a implementação de reuniões diárias, conseguiríamos ver a entrega em um menor período de tempo e por consequência ter maior controle sobre ela.

Nossa visão sobre o projeto

O projeto sofreu por várias modificações causadas por contratempo e empecilhos que inicialmente não conseguimos prever, ou por melhorias que ao implementar observamos ser melhores que nossa abordagem inicial.

Além disso, como a rotina do grupo estava bastante apressada e dispersa, por conta de outras atividades, seja no trabalho ou nas relações pessoais, ocorreu muita falta de comunicação, contudo todos foram proativos em suas partes para que ao final um produto mínimo pudesse ser gerado.

Apesar das dificuldades e da falta de experiência em projetos ágeis, foi desafiador e muito semelhante a um *workflow* de uma empresa, agregando como uma experiência de como será o mercado de trabalho e nossas funções diante dela.