Universidade Federal Fluminense

Aplicativo de Caronas UFFluir

Segundo Trabalho de Engenharia de Software II

Prof: Leonardo Gresta Paulino Murta

Grupo 7: Laion Corcino
Matheus Marques
Sergio Herman
Thiago Serra

Introdução

O objetivo deste trabalho é mostrar a evolução e contratempos da equipe, evidenciando as experiências práticas que contribuíram para o desenvolvimento pessoal de cada membro.

Sumário

- Controle de Versões
- Controle de Modificações
- Estratégia de Ramificação adotada
- Repositório Remoto
- Testes de unidade, integração, sistema e aceitação
- Dados de monitoramento e controle
- Versão final do produto

Controle de Código

Repositório Remoto - GitHub

- Facilita o compartilhamento de códigos
- Cada dev altera uma parte do código localmente
- Resolve modificações que podem gerar conflitos
- Guarda um histórico de versões
- Ajuda a entender a linha do tempo do código

Controle de Versões - Git

Uso de tags

- Facilitam o rastreamento de versões específicas e a recuperação de estados anteriores do repositório
- Marcam pontos de lançamento de software, permitindo que desenvolvedores e usuários voltem a versões estáveis conhecidas do código
- Diferente de branches que podem continuar a receber commits, as tags são normalmente imutáveis

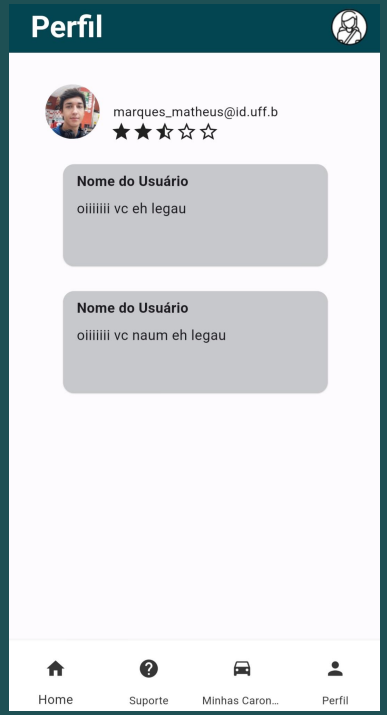
Controle de Versões - Git

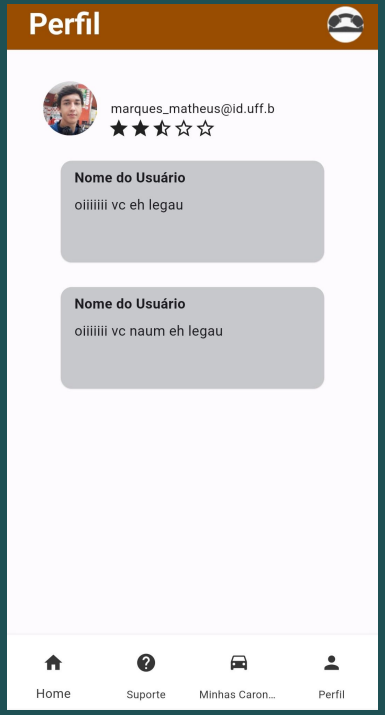

```
git tag -a v0.5 -m "Front end pronto, ainda sem integração com o back (firebase)"
git push origin v0.5

git tag -a v0.8 -m "Demo: principais funcionalidade funcionado (front e back)"
git push origin v0.8
```

Refatoração → v0.8.1

Planejamento de Refatoração





Controle de Modificações

Tentamos o pull request, mas…

- Equipe pequena e segmentada
- Pouca experiência com este recurso
- Atrapalhou mais do que ajudou

Estratégia de Ramificações

- **Uma branch por issue: ineficiente**
- **Poucos membros por subdivisão de trabalho**
- **Pastas separadas para back e front, antes da integração**
- **Comunicação frequente após mudança no back-end**
- **Main como branch única de trabalho**

Avanços no Produto

Melhorias de Design

Experiência do Usuário

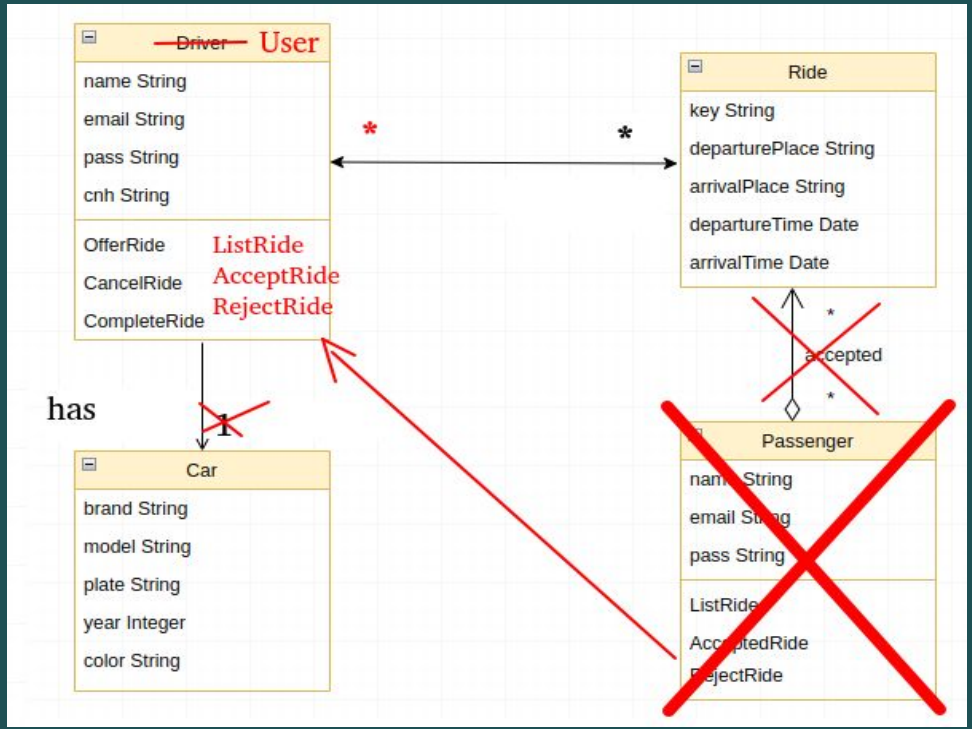
A partir de feedbacks pudemos melhorar o design para tornar o aplicativo mais **intuitivo** e com aparência mais **harmoniosa**, permitindo **fácil identificação** dos componentes e funcionamento, evitando o usuário confundir se está no **escopo** do motorista ou do passageiro.

A **coleta de feedbacks** se deu a partir de **relatos** do usuário usando uma **simulação do Figma**, narrando suas expectativas ao usar cada componente.

Versão Inicial do Back-End

- ➔ Conclusão da Oferta de Caronas
- ➔ Implementação e Testes
- ➔ Esbarramos no deploy

Alterações na UML



Mudança para Firebase

- **Dados não seriam acessíveis pelo celular**
- **Dados não estariam disponíveis online**
- **Falta de mão de obra**
- **Fácil integração com o Dart e Flutter**
- **Dados na nuvem**
- **Gratuito**
- **Unificação de front e back**

Sem Firebase

- Dados locais
- Dados inacessíveis para dispositivos móveis
- Integração complexa

Com Firebase

- Dados em servidor
- Dados disponíveis para dispositivos móveis
- Integração simples

Testes

Testes

Foram criados testes unitários e de integração para a API uffluir.

- ➔ Unitários

  Forma de teste de software onde pequenas partes individuais do código, geralmente funções ou métodos, são verificadas isoladamente. O objetivo é garantir que cada unidade de código funcione conforme esperado.

- ➔ Integração

  Verificam a interação entre diferentes módulos ou componentes de um sistema. O objetivo é assegurar que os componentes funcionem corretamente quando combinados.

Testes Unitários camada de serviço

```java
@Test  Laion Corcino
@DisplayName("deve_aceitar_corrida_com_sucesso")
public void acceptRide() {
    AcceptRequest acceptRequest = buildAcceptRequest();
    Passenger passenger = buildPassenger();
    Ride ride = buildRide();

    Mockito.when(passengerRepository.findByEmail(eq( value: "passenger@id.uff.br"))).thenReturn(passenger);
    Mockito.when(rideRepository.findById(eq( value: 1L))).thenReturn(Optional.of(ride));
    Mockito.when(rideRepository.save(any(Ride.class))).thenReturn(ride);

    Ride acceptedRide = rideService.acceptRide(acceptRequest,  rideId: 1L);

    assertThat(acceptedRide).isNotNull();
    assertThat(acceptedRide.getPassengers()).contains(passenger);
}
```

Testes Unitários camada web

```java
@Test  Laion Corcino
@DisplayName("deve_aceitar_corrida_com_sucesso")
public void acceptRide() throws Exception {
    AcceptRequest acceptRequest = buildAcceptRequest();
    Ride ride = buildRide();

    Mockito.when(rideService.acceptRide(any(AcceptRequest.class), eq( value: 1L))).thenReturn(ride);

    MockHttpServletRequestBuilder putRequest = doPut( path: "/accept/1", JsonUtil.toJson(acceptRequest));

    mockMvc.perform(putRequest)
            .andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$.rideId").value(ride.getRideId()))
            .andExpect(jsonPath( expression: "$.status").value( expectedValue: "OPEN"));
}
```

Testes de integração

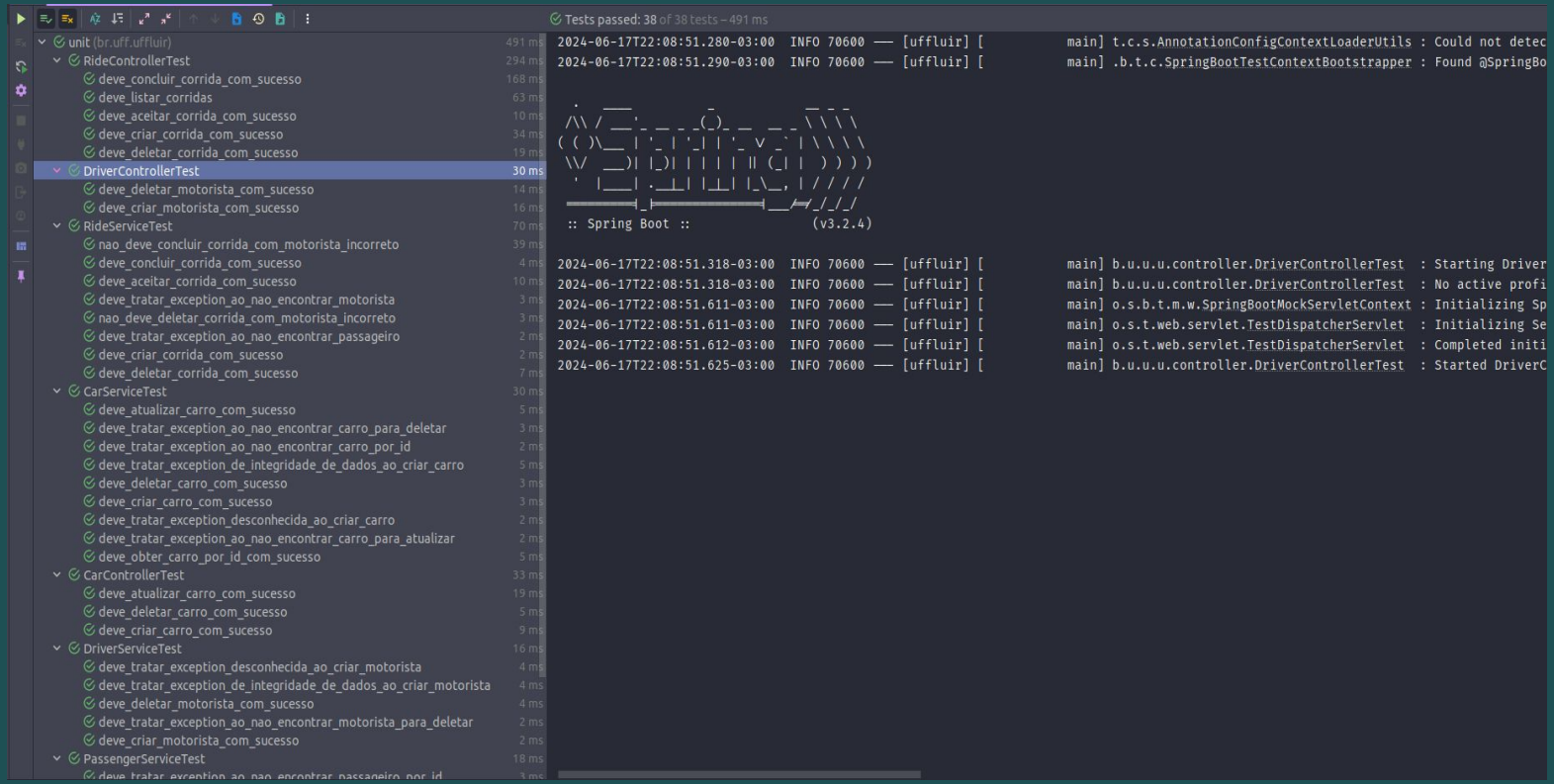
```java
@Test  Laion Corcino
@DisplayName("deve_criar_corrida_com_sucesso")
public void createRide() {
    RideRequest rideRequest = new RideRequest(
            mail: "driver@id.uff.br",
            departurePlace: "Instituto de Computação",
            arrivalPlace: "Campus Gragoatá",
            departureTime: "2023-06-20 10:00:00",
            arrivalTime: "2023-06-20 10:30:00",
            size: 4
    );

    ResponseEntity<RideResponse> postResponse = doPostRide(rideRequest);

    assertThat(postResponse).isNotNull();
    assertThat(postResponse.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    assertThat(postResponse.getBody()).isNotNull();
    assertThat(postResponse.getBody().getDriver().getEmail()).isEqualTo( expected: "driver@id.uff.br");
    assertThat(postResponse.getBody().getDeparturePlace()).isEqualTo( expected: "Instituto de Computação");
    assertThat(postResponse.getBody().getArrivalPlace()).isEqualTo( expected: "Campus Gragoatá");
    assertThat(postResponse.getBody().getDepartureTime()).isEqualTo( dateTimeAsString: "2023-06-20 10:00:00");
    assertThat(postResponse.getBody().getArrivalTime()).isEqualTo( dateTimeAsString: "2023-06-20 10:30:00");
    assertThat(postResponse.getBody().getSize()).isEqualTo( expected: 4);
}
```

Testes executados



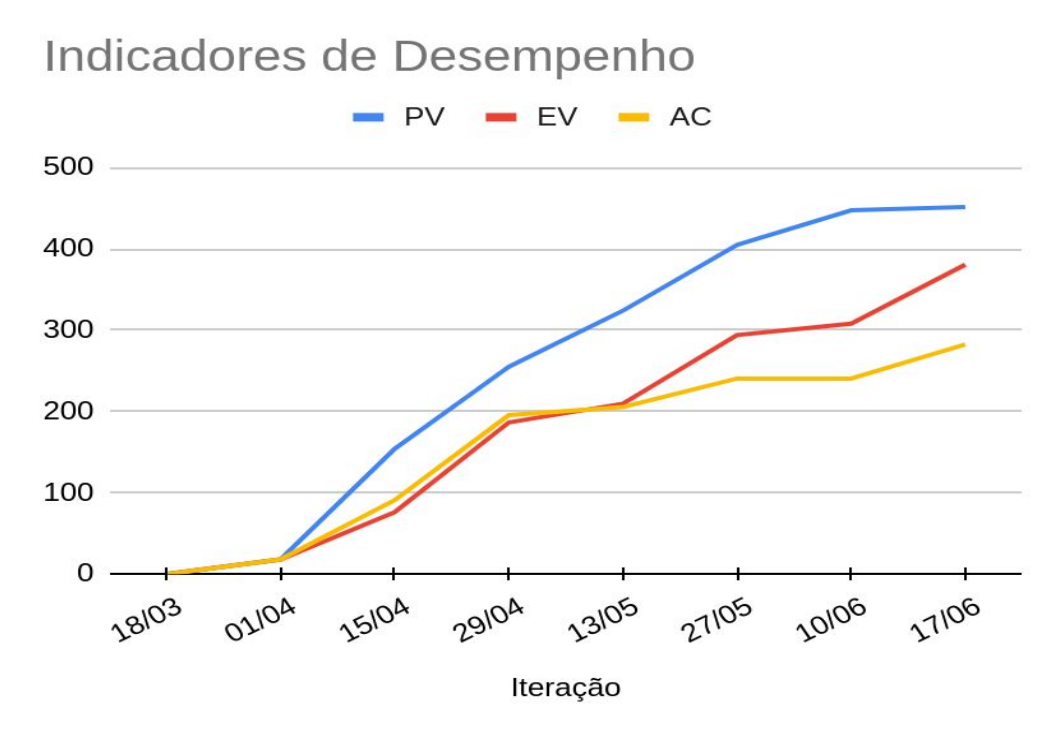
Monitoramento e Controle

Monitoramento e Controle

O projeto está atrasado (SPI < 1), porém abaixo do orçamento CPI > 1).

Data Fim	Iteração	PV	EV	AC	SPI	SV	CPI	CV
01/04	**1**	18	18	18	1,0	0,0	1,0	0,0
15/04	**2**	154	76	91	0,5	-78,0	0,8	-15,0
29/04	**3**	255	186,5	195,5	0,7	-68,5	1,0	-9,0
13/05	**4**	324	209,5	205,5	0,6	-114,5	1,0	4,0
27/05	**5**	405	294	240,5	0,7	-111,0	1,2	53,5
10/06	**6**	447,5	308	240,5	0,7	-139,5	1,3	67,5
17/06	**7**	451,5	380,5	282,5	0,8	-71,0	1,3	98,0

Monitoramento e Controle



Obstáculos Enfrentados Pela Equipe

Muitas alterações ao longo do desenvolvimento

Back-end usando Firebase

Linguagem Java para Dart

Reestruturação das Classes

Problemas

Equipe inexperiente

Falta de comunicação

Saída de membros

Risco 2	
Descrição	Duas pessoas do grupo saem da disciplina
Probabilidade	4%
Impacto	0,4
Exposição	0,02
Prioridade	Baixa

Mitigação 1 a 5	
Tipo	Contenção
Descrição	Oferecer suporte aos colegas tanto nesta disciplina como nas demais
Tipo	Contingência
Descrição	Não deixar uma tarefa sendo responsabilidade de apenas uma pessoa

Risco 7	
Descrição	Uma pessoa fica indisposta por uma semana
Probabilidade	60%
Impacto	0,1
Exposição	0,06
Prioridade	Baixa

Mitigação 7	
Tipo	Contingência
Descrição	1. Verificar se a dupla pode assumir todo o trabalho da equipe temporariamente 2. Dividir o trabalho com a equipe menos atarefada no momento

Situações não Previstas no Gerenciamento de Risco

Greve e trancamento especial aumenta probabilidade de saída

Indisposição prolongada de membros ativos

Soluções

Priorização de atividades essenciais

Redistribuição de tarefas e responsabilidades

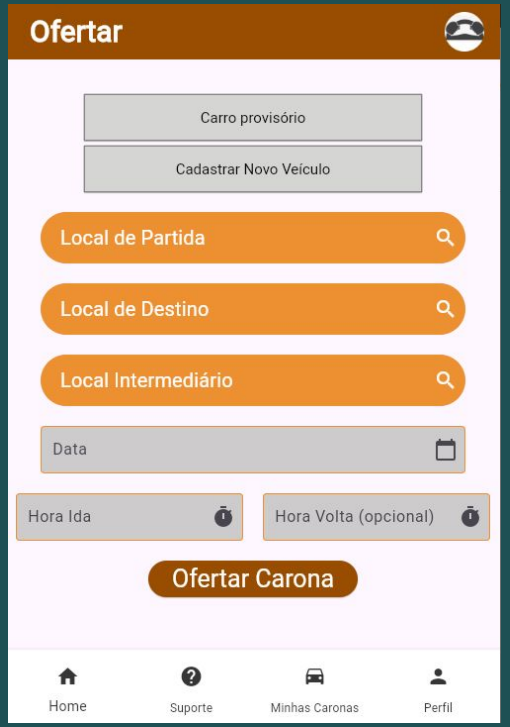
Soluções Possíveis Apenas no Ambiente Empresarial

Motivação da equipe com bonificação de acordo com o desenvolvimento

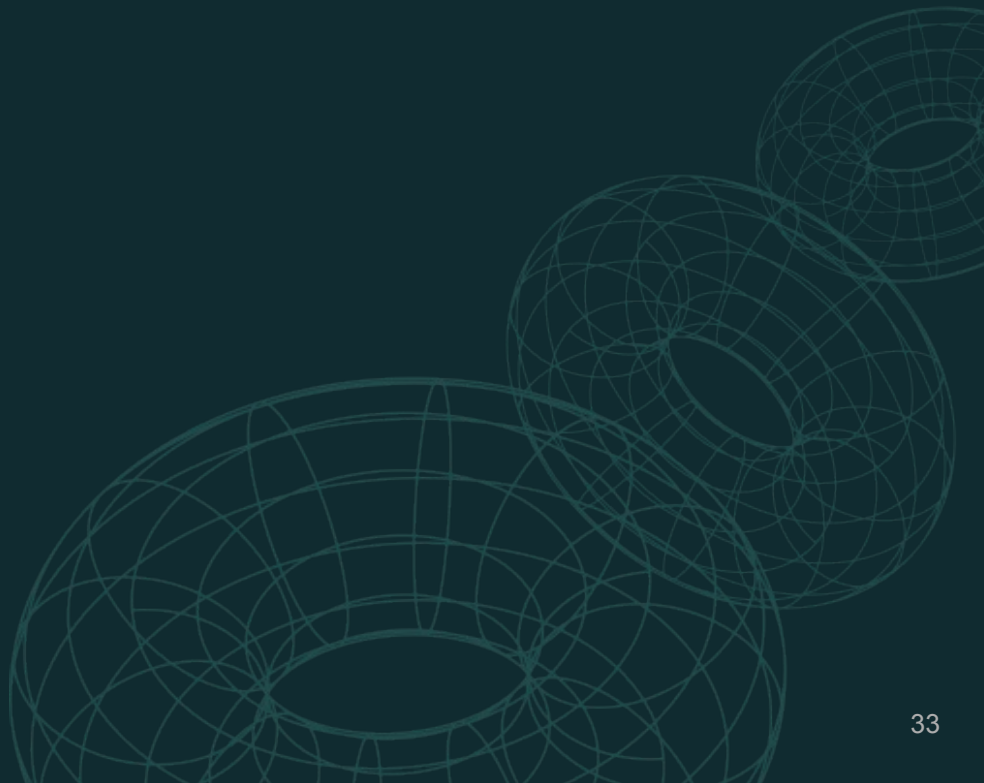
Contratação de desenvolvedores experientes e comunicativos

Versão Final do Produto

Produto Final



Perguntas?



Universidade Federal Fluminense

Aplicativo de Caronas UFFluir

Segundo Trabalho de Engenharia de Software II

Prof: Leonardo Gresta Paulino Murta

Grupo 7: Laion Corcino
Matheus Marques
Sergio Herman
Thiago Serra