

UNIVERSIDADE FEDERAL FLUMINENSE  
INSTITUTO DE COMPUTAÇÃO



# RIDE UFF

Gerência e Manutenção do Software

BRUNA MORAES / BRUNO MELLO / JULIA RIBAS /  
LUIS ADRIANO / THAIS MOYLANY

# Tópicos



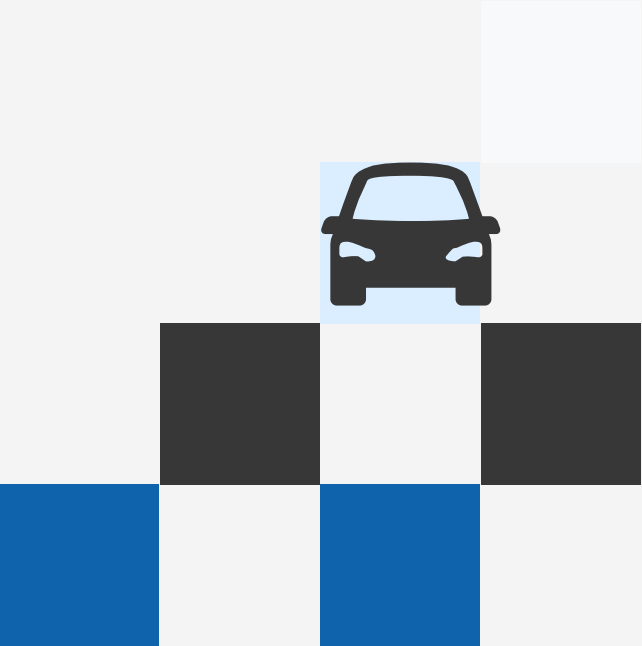
- Escopo do Produto – Requisitos
- Escopo do Projeto – EAP
- Estimativa de Esforço
  - Planning Poker
  - APF
- Custo e Orçamento
- Cronograma de desenvolvimento (GANTT)
- Análise de Riscos:
  - Probabilidade X Impacto
  - Plano de Contenção e Contingência
- Monitoramento e controle
  - Burndown
  - Análise de Valor Agregado
- Versão Parcial do Produto – demo

# Escopo do Produto – Requisitos



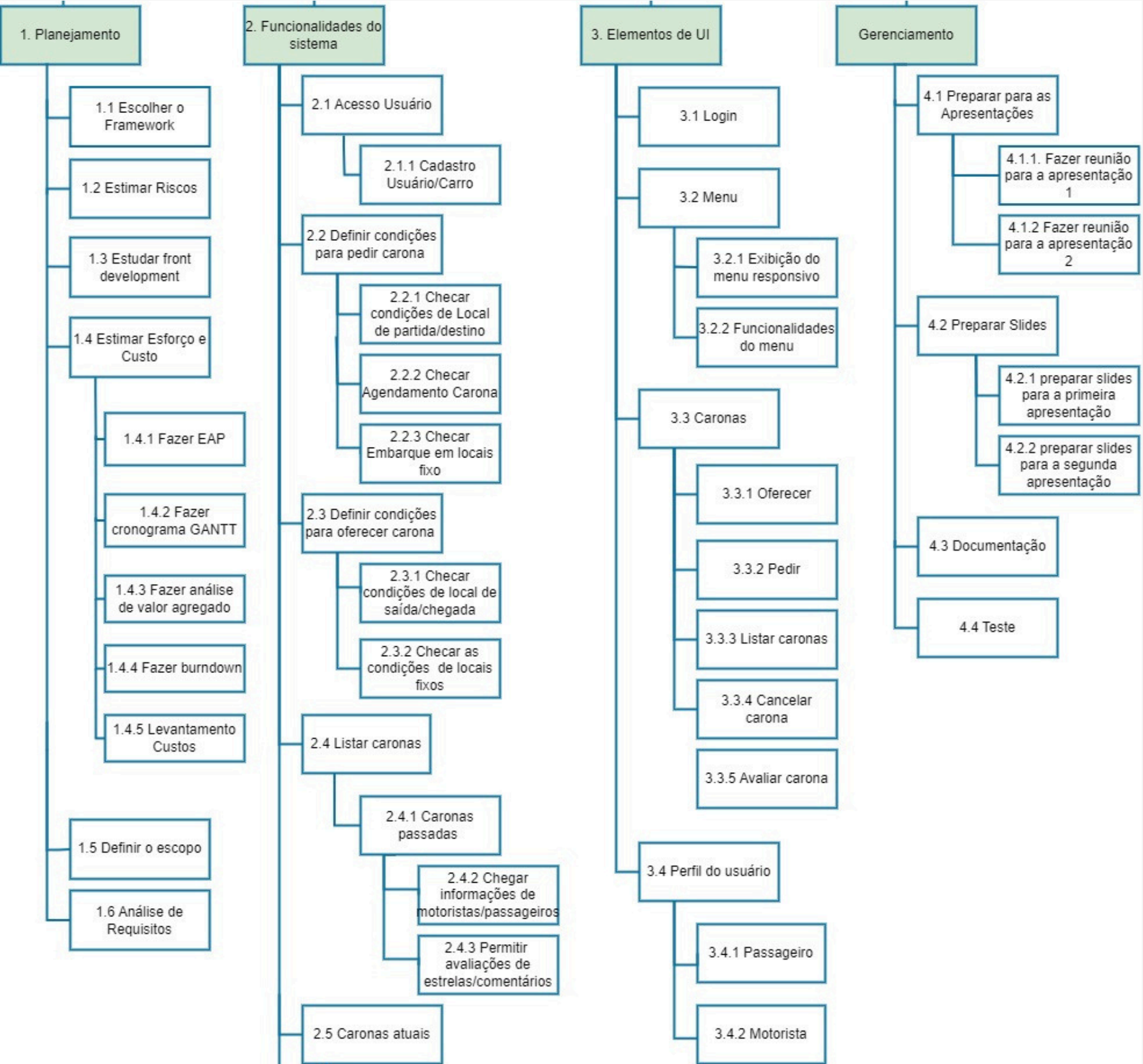
[Acesse aqui](#)

RF - Autenticação de Usuário	RF - Ver Perfil	RF - Pedir Carona	RF - Oferecer Carona	RF - Listar Caronas
Os usuários devem se autenticar ao entrar no aplicativo para acessar suas funcionalidades.	Os usuários podem visualizar e atualizar seus dados, como veículos cadastrados, e verificar sua reputação no sistema.	Permitir aos usuários solicitar uma carona, informando o local de partida, destino, data, hora da viagem e quantidade de passageiros desejada:	Usuários podem oferecer caronas, fornecendo detalhes como local de saída, chegada, data, hora e quantidade de passageiros que podem levar.	Permite aos usuários visualizar informações sobre caronas passadas e atuais, indicando quem foram motoristas ou passageiros em cada carona.
		O sistema oferece opções de locais fixos (como campi da universidade, terminais rodoviários, shoppings) e locais escolhidos pelo usuário através de um mapa.	Os usuários podem escolher entre aceitar corridas automaticamente ou manualmente.	Para caronas passadas, é possível avaliar o motorista e os passageiros presentes através de estrelas e comentários.
		Os usuários podem optar por embarque imediato ou agendado.	Há a opção de oferecer o retorno, informando o horário.	Caronas atuais têm opção de um chat com o grupo da viagem para cancelar a viagem, além do motorista poder aceitar manualmente as solicitações de carona.
			Caso seja primeiro acesso, o usuário precisará cadastrar dados do veículo	

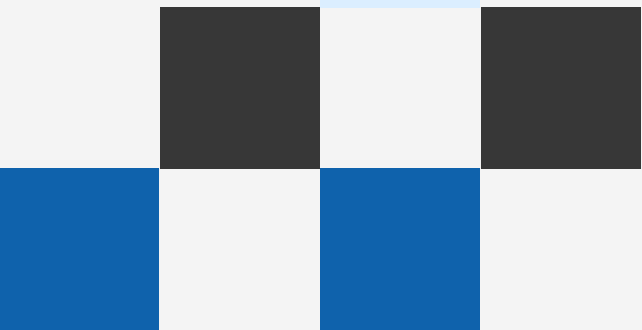




# Escopo do Projeto – EAP



**Acesse aqui**



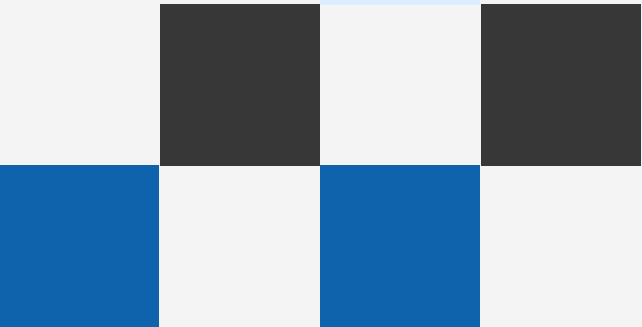
# Estimativa de Esforço – Planning Poker

TOTAL DE HORAS ESTIMADAS:  
209,5

	Código	Nível 1	Nível 2	Nível 3	Nível 4	Nível 5	Nível 6	Nível 7	Estimativa (h) Planning Poker	Valor real (h)
1	1.1	Planejamento	Alinhamentos	Escolher Framework					2,00	2 (nível3)
2	1.2	Planejamento	Estimar riscos						3,50	2 (nível2)
3	1.3	Planejamento	Estudar Front Development						10,00	10
4	1.4	Planejamento	Estimar esforço e custo	EAP	Cronograma Gantt	Análise de Valor Agregado	Burndown	Elaborar esforço e Custos	18,00	20 (todos)
5	1.3.3	Planejamento	Definir o escopo						1,50	2
6	1.4.1	Planejamento	Análise de requisitos						3,00	4
7	2.1	Funcionalidades do Sistema	2						6,00	5 (nivel 2)
8	2.2	Funcionalidades do sistema	Definir condições para pedir carona	Checar condições de Local de partida/destino	Checar Agendamento Carona	Checar Embarque em locais fixo			20,00	7 (nivel 2)
9	2.3	Funcionalidades do sistema	Definir condições para oferecer carona	Checar condições de local de saída/chegada	Checar as condições de locais fixos	Checar retorno ao confirmar corridas			22,00	7 (nivel 2)
10	2.4	Funcionalidades do sistema	Listar caronas	Caronas passadas	Chegar informações de motoristas/passageiros	Permitir avaliações de estrelas/comentários			7,00	7 (nivel 2)
11	2.5	Funcionalidades do sistema	Caronas atuais	Permitir chat com o grupo da viagem	Permitir cancelamento da viagem	Permitir o motorista aceitar a viagem manualmente	Permitir o motorista aceitar a viagem automaticamente	Permitir o motorista concluir a viagem	35,00	2 (nivel 1)



Acesse aqui





# Estima de Esforço – APF

Elementos do Software	Entidades	Campos	Complexidade	Peso
Números de Entradas Externas (EE)	3	18	Alta	6
Números de Saídas Externas (SE)	2	8	Média	5
Números de Consultas Externas (CE)	2	3	Baixa	3
Número de Arquivos Lógicos Internos (ALI)	3	3	Baixa	7
Número de Arquivos de Interface Externos (AIE)	1	2	Baixa	5

**APF via COCOMO**  
**Linguagem: Java / linhas de código: 3,180.**  
**Esforço: 8 homens/mês aproximadamente.**  
**Duração : 5 meses (800 horas aproximadamente)**  
**Custo: 24.240,00 reais.**



**Acesse aqui**





# Custo e Orçamento

**Humano: R\$3928.12**

R\$18.75 \* 209.5 homem-hora

**CapEx: R\$2071.95**

R\$9.89 uso-hora (computador) \* 209.5 homem-hora

**Consumo: R\$135**

internet: R\$100 mensal x 1 mês

Google API R\$35 mensal x 1 mês

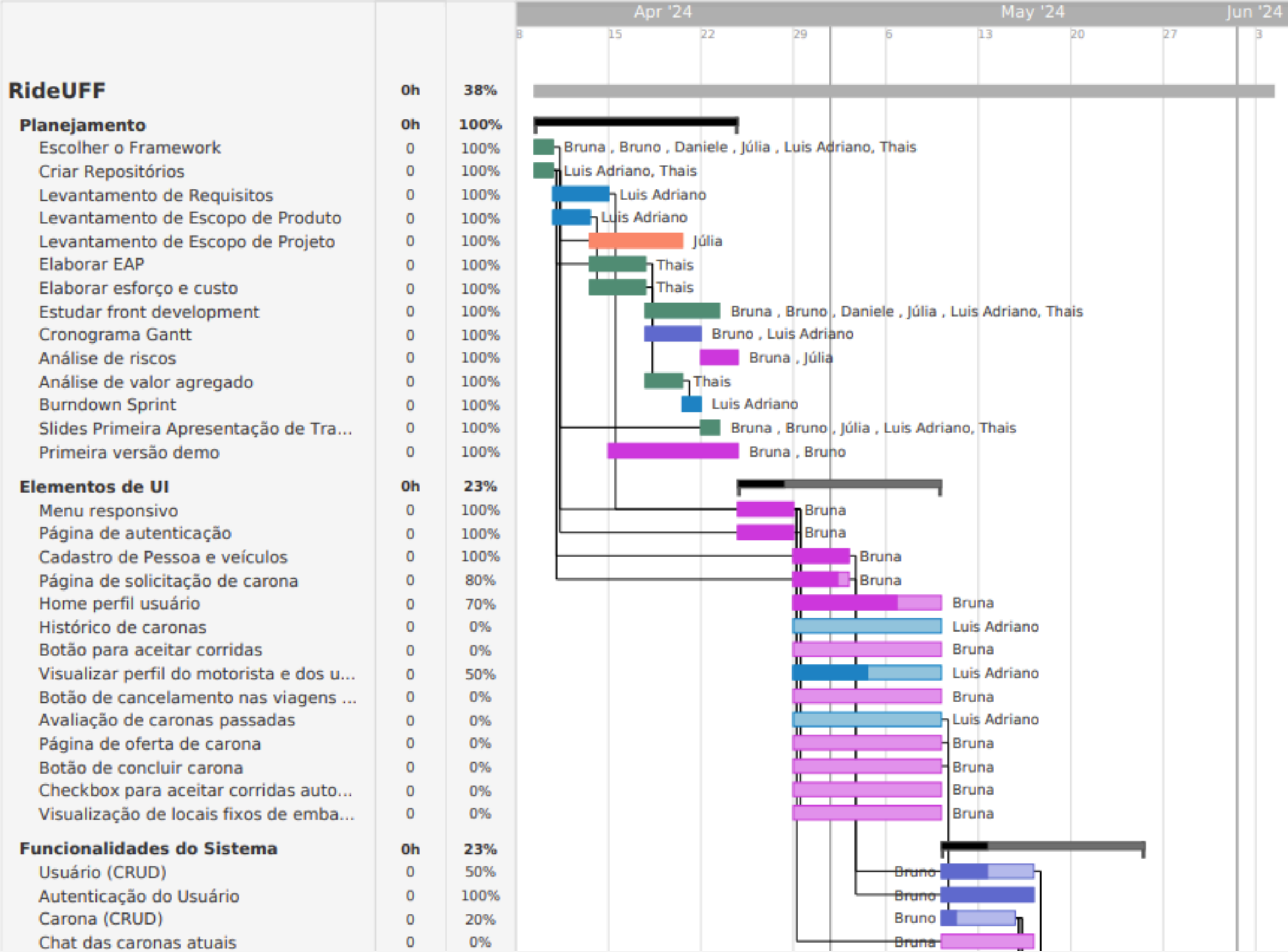
**Custo Total: R\$6135.07**

**Lucro: R\$2454**

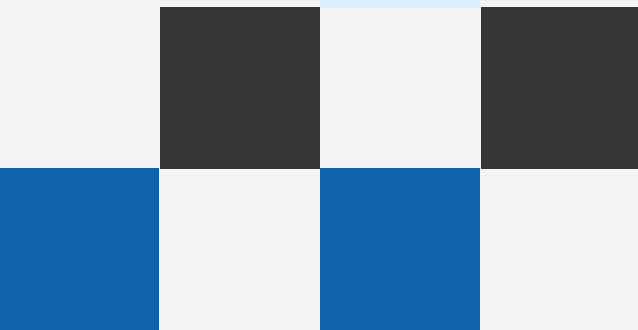
**Orçamento: R\$8589.07**



# Cronograma de desenvolvimento (GANTT)

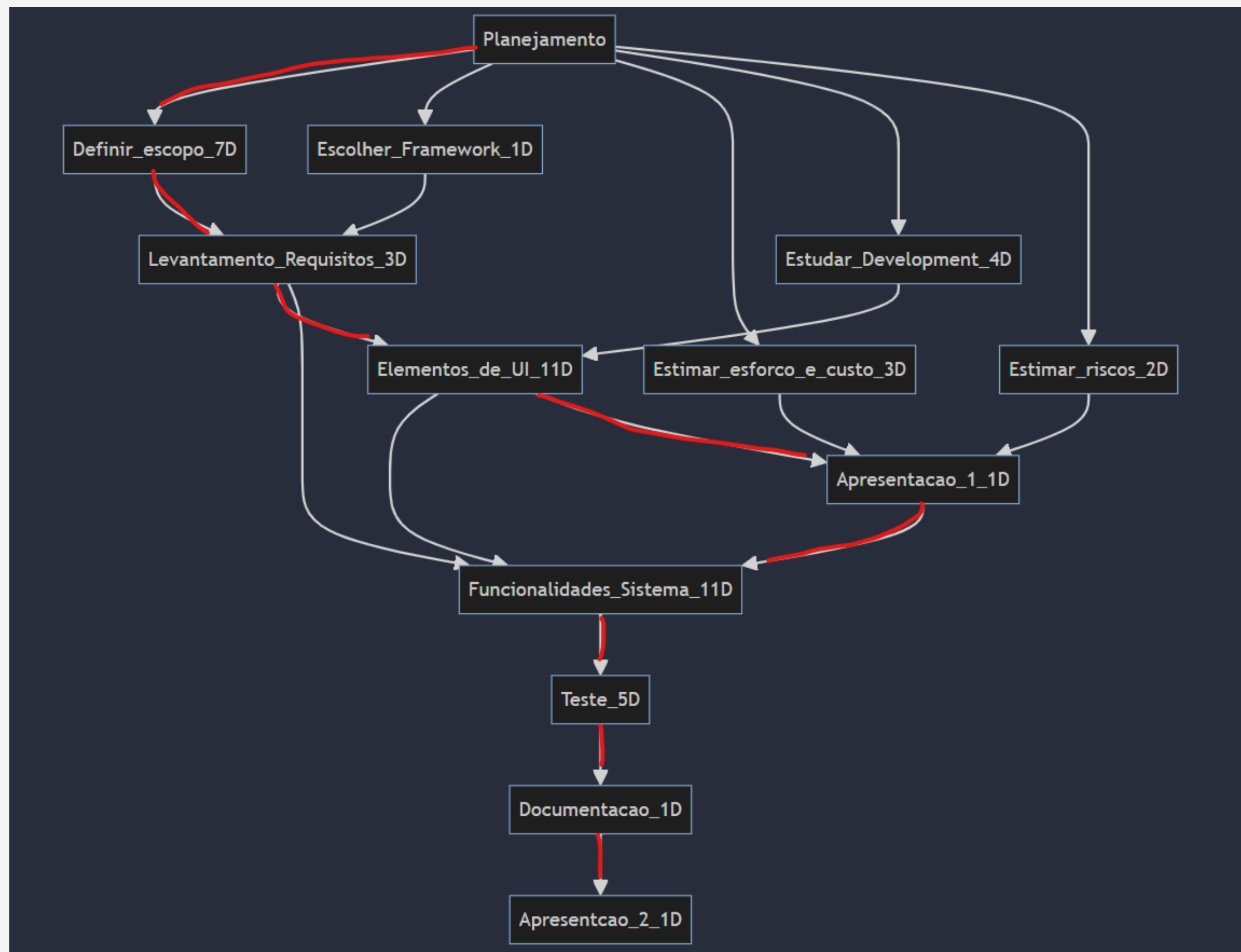


[Acesse aqui](#)



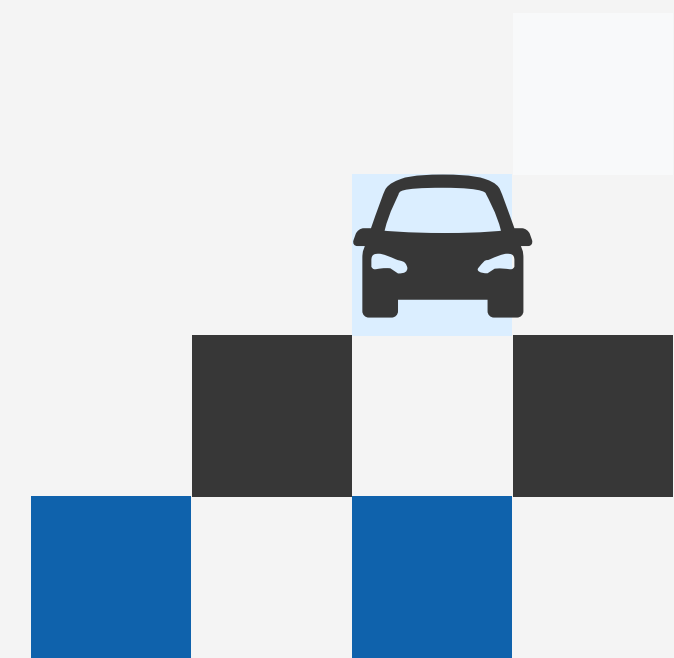


# Caminho Crítico



**Projeto = 40 dias úteis**

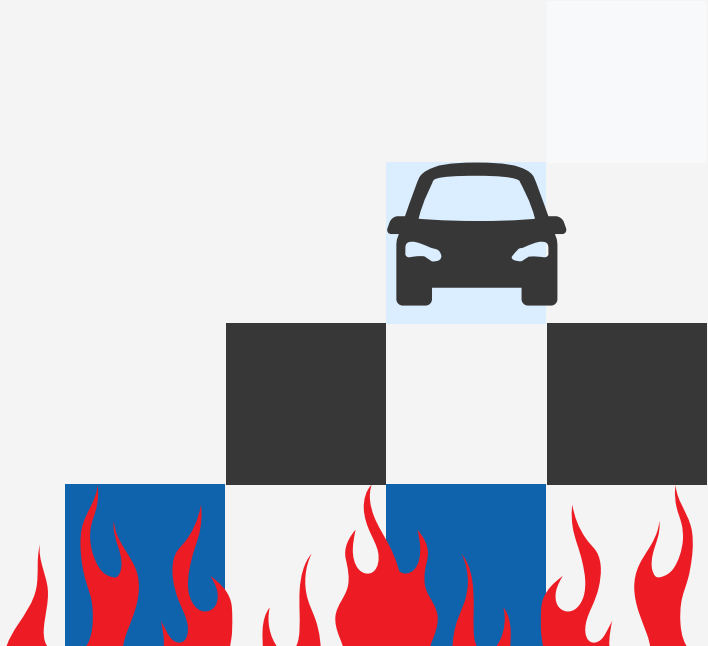
**Caminho Crítico = 40 dias**



# Análise de Riscos

## Probabilidade x impacto Plano de contenção e contingência

Identificação dos Riscos				
Risco	Probabilidade	Impacto	Exposição	Prioridade
Falta de tempo por problemas externos	60%	0,40	0,36	Alta
Dificuldades técnicas no desenvolvimento	70%	0,5	0,36	Alta
Dificuldade em comparecer na reunião	85%	0,2	0,16	Média
Tarefa mal estimada	60%	0,3	0,24	Média
Ausência por doença	20%	0,8	0,16	Média
Problemas técnicos em equipamentos	5%	0,9	0,045	Baixa





# Análise de Riscos

## Probabilidade x impacto Plano de contenção e contingência

1. Falta de tempo por problemas externos

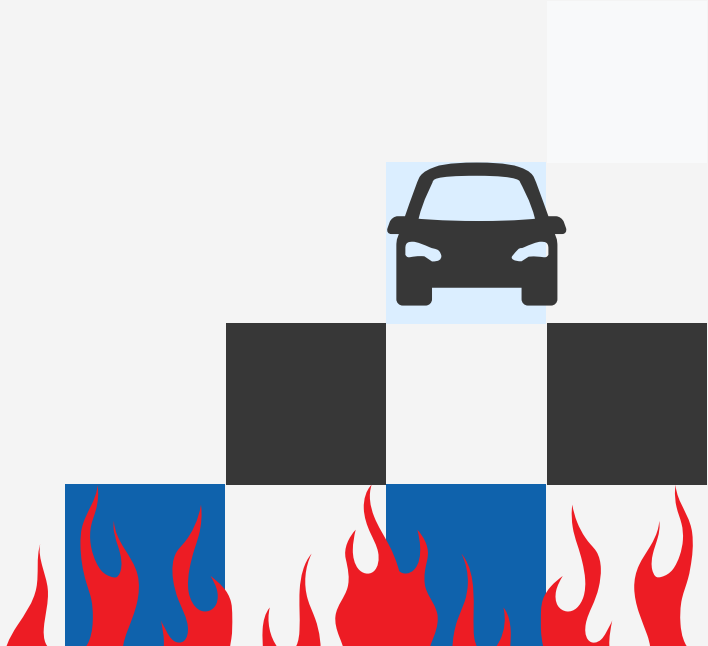
<b>Prob.:</b> 0,6 <b>Impacto:</b> 0,4 <b>Exposição:</b> 0,24	<b>Contenção:</b> Sprints definidas com folgas em dias	<b>Contingência:</b> <u>Agendar reunião em dias standby ou realizar com integrantes presentes.</u>
--	--	---

2. Dificuldades técnicas no desenvolvimento

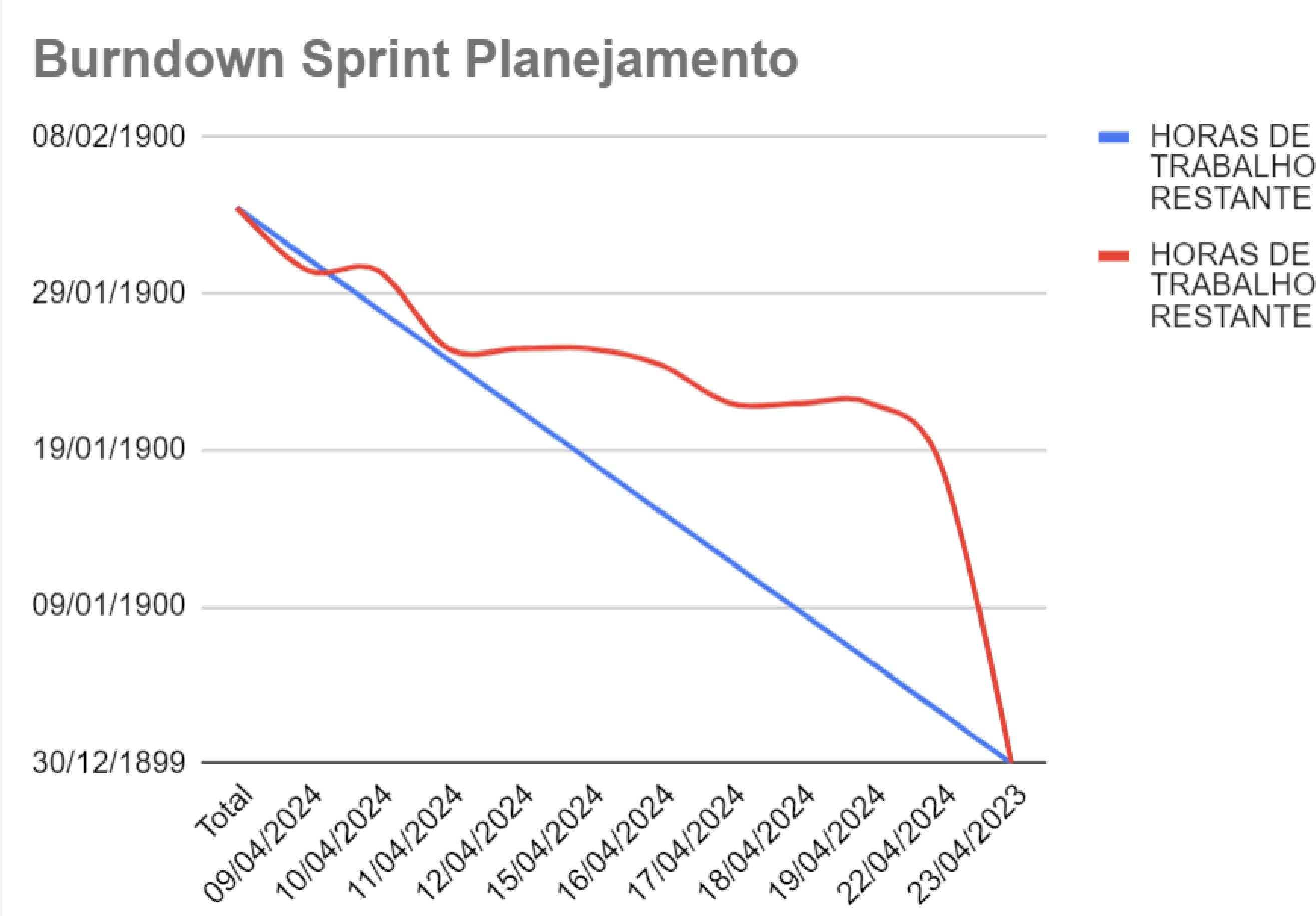
<b>Prob.:</b> 0,7 <b>Impacto:</b> 0,5 <b>Exposição:</b> 0,36	<b>Contenção:</b> Utilizar ferramentas de conhecimento do time	<b>Contingência:</b> Simplificar o escopo do projeto
--	--	---

3. Dificuldade em comparecer à reunião

<b>Prob.:</b> 0,85 <b>Impacto:</b> 0,2 <b>Exposição:</b> 0,16	<b>Contenção:</b> Horários de reuniões pré agendados para standby	<b>Contingência:</b> Gravar reunião e elaborar uma ata para disponibilizar para o time
---	---	--



# Monitoramento e controle – Burndown



[Acesse aqui](#)



# Monitoramento e controle – Análise de valor agregado

Sprint 1		Esforço estimado	Esforço real
Escolher Framework		1	1
Criar Repositórios		1	1
Levantamento de requisitos		3	4
Levantamento de escopo de produto		1	1
Levantamento de escopo de projeto		1	1
Elaborar EAP		3	4
Elaborar esforço e custo		3	4
Estudar front development		10	20
Cronograma Gantt		3	4
Análise de riscos		3,5	2
Análise de valor agregado		3	4
Burndown Sprint		3	4
Slides 1ª apresentação		2,5	5
Sprint 2			
Menu responsivo		4	2,5
Pagina Autenticação		3	2
Cadastro de Pessoa		3	3
Home perfil usuario		4	4
Pagina de solicitação de carona		5	
Historico caronas		5	
Botao (aceitar corrida)		1	
visualizar perfil		6	
botao (cancelar)		5	
avaliação de caronas passadas		5	
pagina pedir carona		3	
botao (concluir carona)		1	
checkbox (aceitar corridas automaticamente)		1	
visualizar locais fixos de embarque		5	

97 horas planejadas até o final da segunda sprint

66.5 horas realizadas até agora)

$$BAC = 8589,07$$

$$PV = 8589,07 * 0,4630 = 3976,74$$

$$P\%C = 46.3\%$$

$$EV = 8589,07 * 0,3174 = 2726,15$$

$$A\%C = 31.74\%$$

$$SPI = 2726,15 / 3976,74 = 0,69 \text{ Atrasados!}$$

$$SV = 2726,15 - 3976,74 = -1250,59$$

$$CPI = 2726,15 / (66,5 * 18,75) = 2,19$$

Mas Abaixo do custo

$$CV = 2726,15 - 1246,88 = 1479,27$$



# Versão Parcial do Produto – Tech Utilizadas

## FRONTEND

HTML  
CSS  
JS  
BOOTSTRAP

## BACKEND

JAVA  
SPRING BOOT

## BANCO DE DADOS

MYSQL



[Acesse aqui o github do projeto](#)







### Login

Usuário:

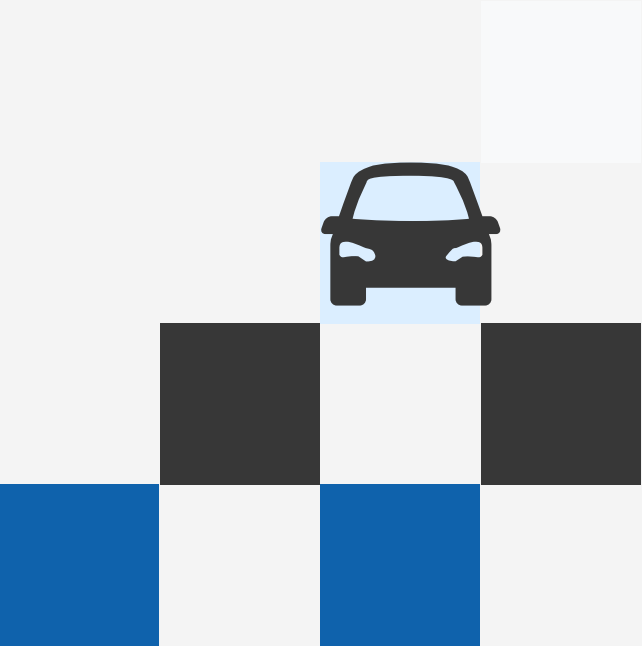
Senha:

☐ Lembrar de mim | [Esqueci a senha](#)

Entrar

Não possui cadastro RIDE UFF ainda?

[Fazer Cadastro](#)



# Versão Parcial do Produto - demo





# Versão Parcial do Produto – demo

RF – Pedir Carona

RideUFF |

Welcome, usuário

Nome:Fulano

Matrícula:789ABC

Pedir Carona

Encontre uma carona rapidamente

Oferecer Carona

Compartilhe sua rota e ajude outros usuários

Listar Caronas

Visualize caronas disponíveis e passadas

Ver Perfil

127.0.0.1:5500/pages/pedir-carona.htmlações pessoais atualizadas

RideUFF |

Home

Sobre

Opções do Usuário ▾

Logout

Pedir Carona

Local de Partida:

Local de Destino:

Tipo de Embarque:

Embarque Agendado ▾


Data da Viagem:

30/04/2024 📅

Hora da Viagem:

08:49 🕒

Buscar Carona

A decorative graphic in the bottom right corner of the slide. It features a black silhouette of a car facing forward, positioned on a light blue square. This square is part of a larger grid of squares in white and blue, arranged in a pattern that tapers to the right.

# Versão Parcial do Produto – demo

RF – Autenticação: Login

```
import org.springframework.beans.factory.annotation.Autowired;

@Validated
@Slf4j
@RestController
public class LoginController {

    @Autowired
    private LoginService loginService;

    @PostMapping(value="/login")
    public ResponseEntity<?> login(

        @RequestBody UsuarioLoginDTO usuario

    ) {

        boolean autenticado = loginService.autenticar(usuario);
        if (autenticado) {
            return new ResponseEntity<>(HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
        }
    }
}
```

Controller

```
@Service
public class LoginService {

    @Autowired
    private AutenticacaoRepository autenticacaoRepository;

    public boolean autenticar(UsuarioLoginDTO usuario) {
        UsuarioLoginDTO usuarioDB = autenticacaoRepository.findByEmail(usuario.getEmail());
        if (usuarioDB != null && usuarioDB.getSenha().equals(usuario.getSenha())) {
            return true;
        } else {
            return false;
        }
    }
}
```

Service

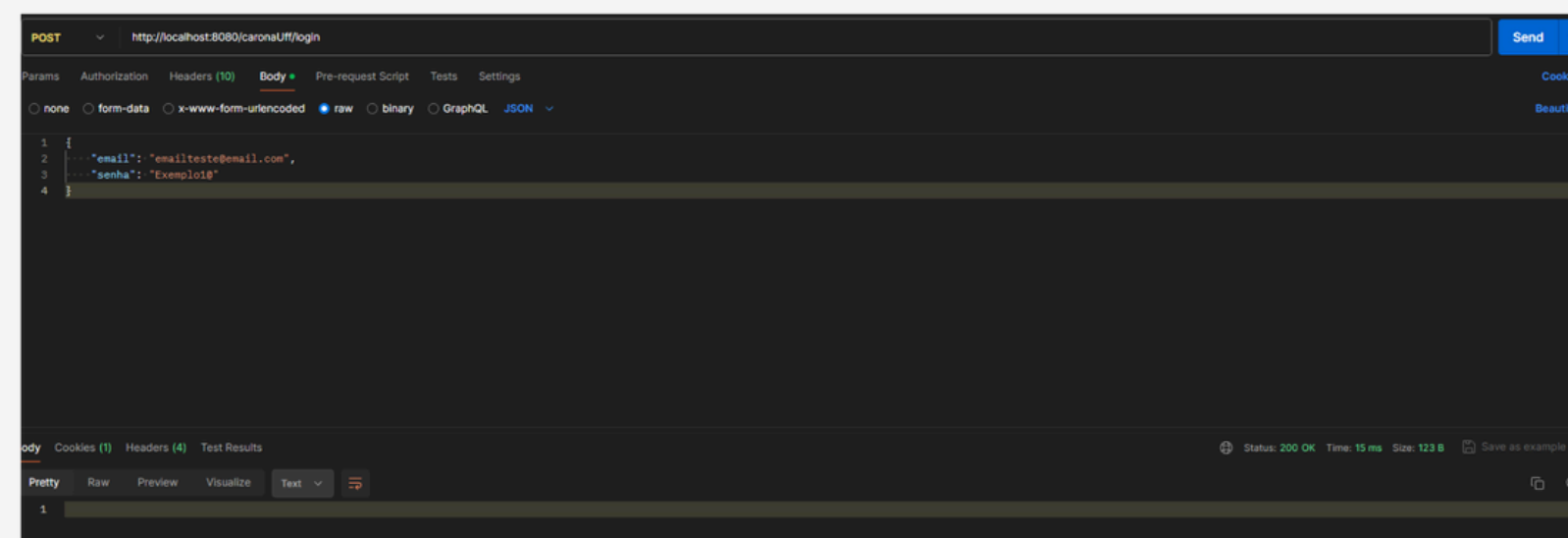
```
@Repository
public class AutenticacaoRepository {

    private static final String QUERY_FIND_BY_EMAIL = "SELECT * FROM usuarios WHERE email = ?";

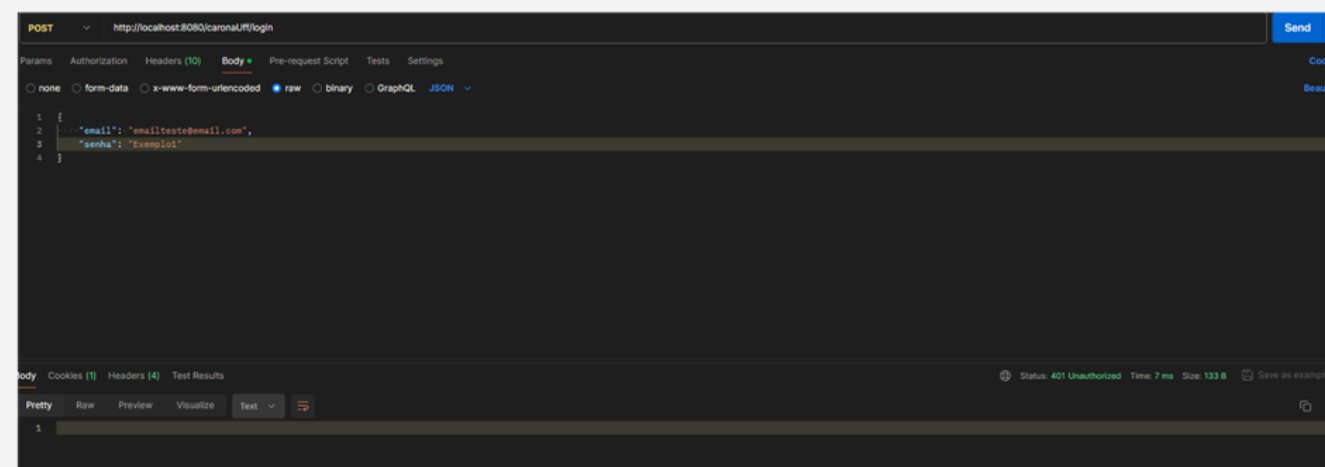
    @Autowired
    private JdbcTemplate jdbcTemplate;

    public UsuarioLoginDTO findByEmail(String email) {
        try {
            return jdbcTemplate.queryForObject(QUERY_FIND_BY_EMAIL, new BeanPropertyRowMapper<>(UsuarioLoginDTO.class), email);
        } catch (EmptyResultDataAccessException e) {
            return null;
        }
    }
}
```

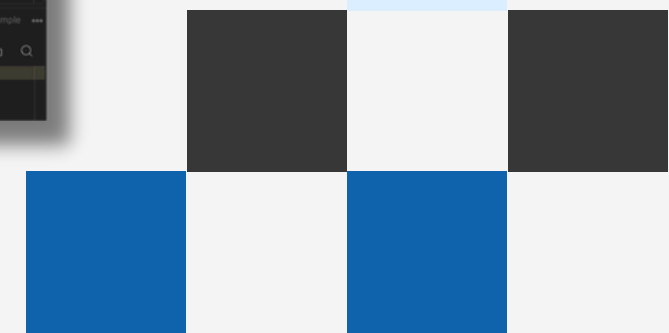
Repository



Chamada Sucesso



Chamada Erro





# Versão Parcial do Produto – demo

```
@Validated
@Slf4j
@RestController
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;

    @PostMapping(value="/criar/perfil")
    public String criarPerfil(

        @RequestBody CriarUsuarioRequestDTO criarUsuarioRequestDTO
    ) {
        Log.debug("criarPerfil() INICIO : criando usuario request={}",criarUsuarioRequestDTO);
        usuarioService.criarUsuario(criarUsuarioRequestDTO);
        Log.debug("criarPerfil() FIM" );

        return "Perfil criado com sucesso";
    }
}
```

## Controller

```
@Slf4j
@Service
public class UsuarioService {

    @Autowired
    private UsuarioRepository usuarioRepository;

    public void criarUsuario(CriarUsuarioRequestDTO usuario) {

        Log.debug("criarUsuario() INICIO : criando usuario usuario={}",usuario);

        if (usuarioRepository.emailExiste(usuario.getEmail())) {
            throw new RuntimeException("Email já cadastrado");
        }

        usuarioRepository.criarUsuario(usuario.getNome(), "0", usuario.getEmail(), usuario.getSenha(), usuario.getPapel(), usuario.getVeiculos().getMarca(), usuario.getVeiculos().getCh);
    }
}
```

## Service

```
@Repository
public class UsuarioRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public boolean emailExiste(String email) {
        SimpleJdbcCall call = new SimpleJdbcCall(jdbcTemplate)
            .withProcedureName("verificar_email")
            .declareParameters(
                new SqlParameter("p_email", Types.VARCHAR),
                new SqlOutParameter("existe", Types.BOOLEAN));

        SqlParameterSource params = new MapSqlParameterSource()
            .addValue("p_email", email);

        Map<String, Object> result = call.execute(params);

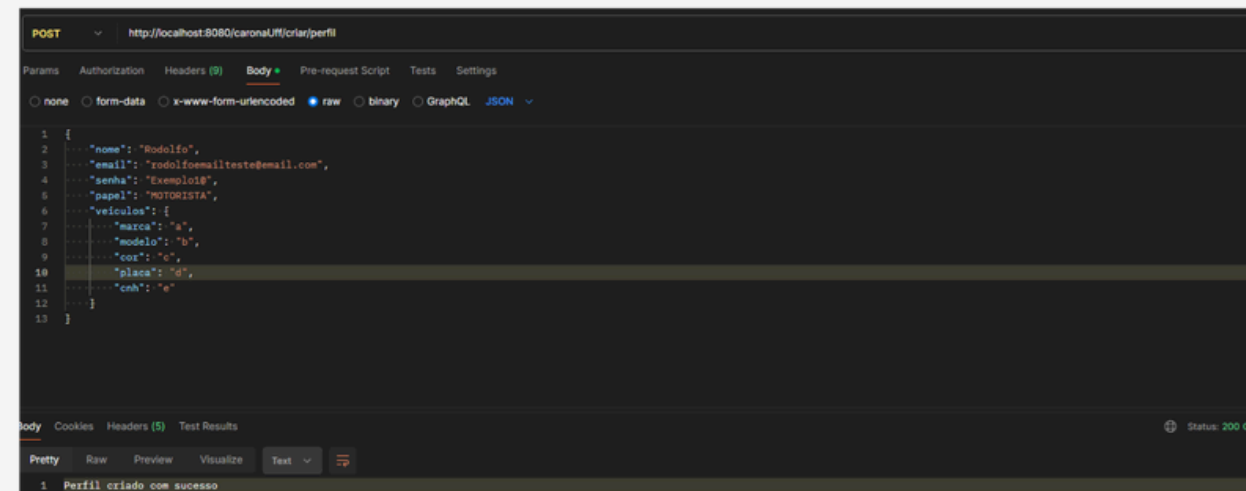
        return (boolean) result.get("existe");
    }

    public void criarUsuario(String nome, String reputacao, String email, String senha, String papel, String marca, String modelo, String cor, String placa, String ch) {
        SimpleJdbcCall call = new SimpleJdbcCall(jdbcTemplate)
            .withProcedureName("criar_usuario");

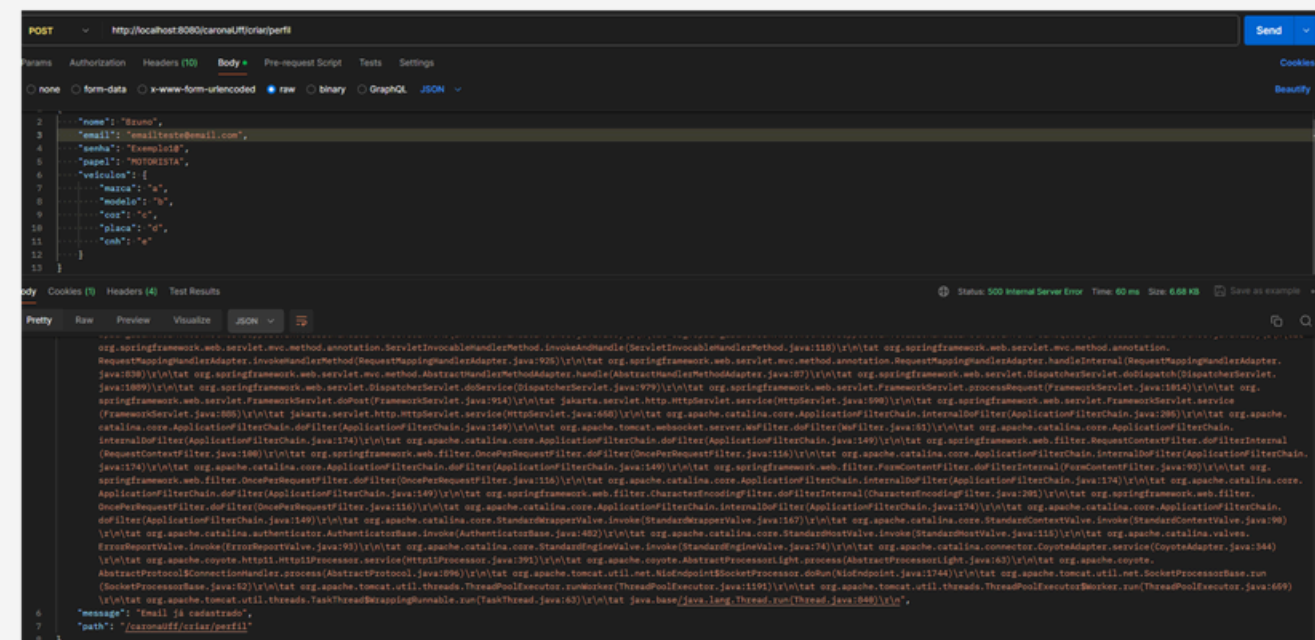
        SqlParameterSource params = new MapSqlParameterSource()
            .addValue("p_nome", nome)
            .addValue("p_reputacao", reputacao)
            .addValue("p_email", email)
            .addValue("p_senha", senha)
            .addValue("p_papel", papel)
            .addValue("p_marca", marca)
            .addValue("p_modelo", modelo)
            .addValue("p_cor", cor)
            .addValue("p_placa", placa)
            .addValue("p_ch", ch);

        call.execute(params);
    }
}
```

## Repository



## Chamada Sucesso



## Chamada Erro



# Versão Parcial do Produto – demo

RF – Tabelas e Procedures

Tables	
> caronas	64K
> carro	32K
> passageiros	32K
> pontosfixos	16K
> usuarios	32K
> veiculos	16K
> viagens	32K
> Views	
> Indexes	
Procedures	
> AtualizarPerfilUsuario	
> criar_usuario	
> ListarCaronas	
> OferecerCarona	
> PedirCarona	
> verificar_email	
> VerPerfilCarona	
> VerPerfilUsuario	
> Triggers	
> Events	

```
CREATE DEFINER='root'@'localhost' PROCEDURE `eventos`.`criar_usuario`(  
  IN p_nome VARCHAR(255),  
  IN p_reputacao INT,  
  IN p_email VARCHAR(255),  
  IN p_senha VARCHAR(255),  
  IN p_papel VARCHAR(255),  
  IN p_marca VARCHAR(255),  
  IN p_modelo VARCHAR(255),  
  IN p_cor VARCHAR(255),  
  IN p_placa VARCHAR(255),  
  IN p_cnh VARCHAR(255)  
)  
BEGIN  
  INSERT INTO usuarios(nome, reputacao, email, senha, papel, marca, modelo, cor, placa, cnh)  
  VALUES (p_nome, p_reputacao, p_email, p_senha, p_papel, p_marca, p_modelo, p_cor, p_placa, p_cnh);  
END
```

