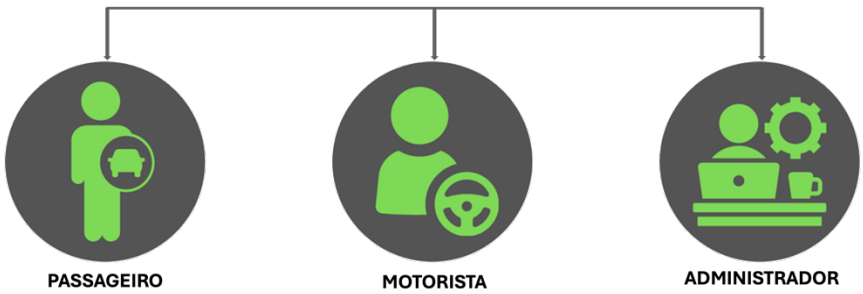
PROJETO – Eng.Sw.2 2025.2

- Este projeto deve ser feito em equipe 3 a 4 alunos.
- Cada grupo deverá criar um repositório no Github para acompanhamento.
    - Pelo repositório será possível acompanhar as contribuições de cada membro do grupo. Se vocês fizerem programação em pares, é importante dividir os commits entre os dois membros alternadamente (um dia faz o commit no nome de um dos alunos e no outro dia faz o commit no nome do outro aluno para termos certeza que todos estão trabalhando).

- **SISTEMA: UberPB (Uber + UEPB)**

  Se trata de um sistema de *mobilidade simplificado*, com base no aplicativo Uber. Onde existem basicamente 3 tipos de usuário (cada um com uma visão diferente do sistema):

  1. Passageiro/Cliente: solicita corridas
  2. Motorista: aceita corridas e transporta passageiros
  3. Administrador: gerencia usuários, pagamentos e suporte



Requisitos Gerais
A seguir temos uma breve descrição requisitos funcionais desejados para o sistema (o que o sistema deve fazer). Observe que, no mundo real, o cliente poderá mudar de idéia a respeito desses requisitos funcionais ao longo do tempo. É até possível que a professora altere os requisitos no meio do semestre. Os requisitos foram priorizados pelo cliente conforme podem ver na descrição das releases mais adiante. Não iremos tratar de requisitos não-funcionais.

Cadastro e Autenticação
RF01: O sistema deve permitir cadastro de passageiros e motoristas.
RF02: O sistema deve validar documentos e características do veículo para definir a categoria em que o motorista pode atuar (UberX, Comfort, Black, etc.).
RF03: O sistema deve permitir login com e-mail.

Solicitação de Corrida
RF04: O passageiro deve poder inserir origem e destino.
RF05: O sistema deve calcular estimativa de tempo e preço com base na categoria de carro escolhida.
RF06: O sistema deve disponibilizar as opções de categorias de veículos:
UberX – Corrida mais econômica.
Uber Comfort – Carros mais novos e espaçosos.
Uber Black – Veículos premium e motoristas de alta avaliação.
Uber Bag – Veículos com porta-malas maior.
Uber XL – Capacidade para mais passageiros.
RF07: O sistema deve notificar motoristas da categoria selecionada pelo passageiro.

Aceite da Corrida
RF08: O motorista deve poder aceitar ou recusar a corrida.
RF09: O sistema deve atribuir a corrida ao motorista mais próximo dentro da categoria solicitada.

Acompanhamento da Corrida
RF10: O passageiro deve acompanhar a localização do motorista.
RF11: O sistema deve atualizar a estimativa de chegada de acordo com a categoria escolhida.
RF12: O motorista deve visualizar a rota otimizada até o destino.

Pagamentos
RF13: O sistema deve permitir pagamento via cartão, PIX, PayPal ou dinheiro.
RF14: O sistema deve calcular automaticamente o valor da corrida de acordo com:
Distância.
Tempo estimado.
Categoria do veículo.
Tarifa dinâmica.
RF15: O sistema deve emitir recibo eletrônico ao final da corrida.

Avaliações
RF16: O sistema deve permitir que passageiros e motoristas se avaliem mutuamente.
RF17: O sistema deve utilizar a média das avaliações para priorizar motoristas (especialmente em categorias premium).

Histórico
RF18: O sistema deve manter histórico de corridas filtrável por categoria de carro.

RF19: Requisito surpresa.

RF_SurpresaN

- **CONSIDERAÇÕES:**
  - Testes: devem ser feitos testes para cada classe com o JUnit e ser calculada a cobertura (mínimo de 80%). Para o Eclipse tem o plugin Emma.
  - Tratamento de erros: deve ser feito tratamento de erros para cada requisito, por exemplo: não deve ser possível cadastrar um usuário que já existe na plataforma,

ou modificar um usuário removido etc. Pensem em todas as possibilidades de erro e deem o devido tratamento.
  - Persistência: o armazenamento deve ser local, ou seja, o sistema deve armazenar em disco todos os dados necessários para manter o seu estado mesmo que o programa seja fechado. Assim, se eu usar o programa, adicionar usuários, corridas, etc. tudo isso deve ser visto se eu fechar o programa e abrir novamente.
  - Interface: vocês vão observar que não pedi interface gráfica, motivo: não quero que você gaste tempo com interfaces gráficas, pois temos outras disciplinas para isso, o foco deve ser na qualidade do projeto. Porém, deve ser feita uma interface de comandos simplificada.
  - Não é para usar API (por ex. Swagger e afins), usar apenas Java, JUnit e padrões de Projeto!

- **SOBRE ENTREGAS:**
  - Cada release deve conter:
    - **Repositório** contendo a implementação dos requisitos relacionado àquela entrega, juntamente com a parte de persistência, a interface simplificada, os testes e o tratamento de erros referentes aos requisitos daquela entrega;
    - Nesse repositório, além do código, crie um pasta chamada "*releases*", lá você deve adicionar o relatório dessa entrega (chamado **relatorio-release**1.pdf), que deve conter:
      (1) A descrição da arquitetura utilizada (diagramas de caso se uso, de classes e de sequência)
      (2) As funcionalidades desenvolvidas e prints do terminal com exemplos das funcionalidades
      (3) Relatório de testes e cobertura de código
    - Nesse repositório, ainda na pasta "*releases*", adicione o relatório de gerenciamento do processo de desenvolvimento dessa entrega (chamado **relatorio-processo-release**1.pdf), que deve conter, para cada sprint:
      (1) O backlog com os itens de trabalho e as tasks;
      (2) A descrição da organização da equipe, os papéis de cada membro;
      (3) Gráfico de burndown de cada sprint, o Azure gera esse gráfico, basta fazer um print e colocar no relatório.

- **AVALIAÇÃO**
  - O projeto será avaliado de acordo com os seguintes pesos:
    - Compilação (0% - se não compilar, o projeto não deve ser aceito)
    - Repositório + apresentação das funcionalidades + testes (30%)
    - Relatório de release (20%)
    - Relatório de gerenciamento do processo de desenvolvimento (50%)

- **CRONOGRAMA**

  - **Release1 – 22/09/2025**
    - RF01 a RF12

- **Release2 – 20/10/2025**
  - RF13 a RF19

- **Release3 – 17/11/2025**
  - Troca de projetos
  - Novos requisitos

- **CRONOGRAMA DE SPRINTS**
  - SPRINT1: 26/08 - 08/09
  - SPRINT2: 09/09 - 22/09
  - SPRINT3: 23/09 - 06/10
  - SPRINT4: 07/10 - 20/10
  - SPRINT5: 21/10 - 03/11
  - SPRINT6: 04/11 - 17/11

- **CRONOGRAMA DE REUNIÕES**
  - SPRINT PLANNING – **26/08**
  - SPRINT REVIEW/PLANNING – **09/09**
  - SPRINT REVIEW/PLANNING – **23/09**
  - SPRINT REVIEW/PLANNING – **07/10**
  - SPRINT REVIEW/PLANNING – **21/10**
  - SPRINT REVIEW/PLANNING – **04/11**
  - SPRINT REVIEW – **18/11**

- **EQUIPES**
  - GRUPO1:
    -
  - GRUPO2:
    -
  - GRUPO3:
    -
  - GRUPO4:
    -
  - GRUPO5:
    -
  - GRUPO6:
    -