



# FIAP GRADUAÇÃO

# CHALLENGE 2025

2º ANO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

*Turmas de Fevereiro – 2º Semestre*

# CRONOGRAMA

2º SEMESTRE

# CRONOGRAMA

DATA	EVENTO	STEAKHOLDER
26/08/2025*	Mentoria com os Professores	Professores
08/09/2025*	Apresentação para Banca de Professores – Seleção dos Projetos	PROFESSORES
Semana do 22/09*	Apresentação dos selecionados para Banca Final Mottu	Mottu
28/09/2025	ENTREGA DA SPRINT 3	ALUNO
05/10/2025	Feedback das entregas SPRINT 3	PROFESSORES
09/11/2025	ENTREGA DA SPRINT 4	ALUNO
16/11/2025	Feedback das entregas SPRINT 4	PROFESSORES
08/11/2025	NEXT	FIAP

\* As datas poderão ser alteradas

# APRESENTAÇÕES – 2º SEMESTRE

- **A primeira apresentação para a banca de professores** têm como objetivo fornecer orientações e direcionamentos para o desenvolvimento dos projetos.
- **A segunda apresentação para a banca de professores** será voltada à avaliação dos projetos que serão **selecionados para a apresentação final à Mottu**.
- **A apresentação final para a Mottu** servirá para selecionar os melhores projetos, que participarão do evento NEXT e concorrerão à premiação na competição.

# 3º ENTREGAS

POR DISCIPLINA

# ADVANCED BUSINESS DEVELOPMENT WITH .NET

- Criar uma API RESTful utilizando .NET (Web API ou Minimal API) com foco em boas práticas REST
  - 25 pts - Mínimo 3 entidades principais (ex: Produtos, Usuários, Pedidos), Justificando a escolha do domínio
  - 50 pts - Endpoints CRUD para as 3 entidades com boas práticas REST (paginação, Hateoas, status code adequados)
  - 15 pts: Configurar Swagger/OpenAPI com:
    - Descrição de endpoints e parâmetros
    - Exemplos de payloads
    - Modelos de dados descritos

# ADVANCED BUSINESS DEVELOPMENT WITH .NET

- 10 – pts Repositório público no GitHub contendo:
  - Código-fonte da API
  - README.md com:
    - Nomes dos integrantes
    - Justificativa da arquitetura
    - Instruções de execução da API
    - Exemplos de uso dos endpoints
    - Comando para rodar os testes



# ADVANCED BUSINESS DEVELOPMENT WITH .NET

- Penalidades
  - -20pts – Falta de documentação do Swagger
  - -100pts – Projeto não compilar
  - -20pts – Sem documentação do Readme no Github

# COMPLIANCE, QUALITY ASSURANCE & TESTS

- ITENS OBRIGATÓRIOS:
  - Desenvolva o plano detalhado do seu projeto Challenge, usando o AZURE BOARDS, contendo:
  - Estrutura de trabalho em forma de Backlog de Produto, com detalhes de Épicas, Features e Product Backlog Itens, no padrão Scrum (peso 20%);
  - Documentação de descrição e critérios de aceite, explicando as histórias de usuário formadas pelos Épicas, Features e Product Backlog Itens (peso 20%);
  - Apontamento da Prioridade, Esforço, Dependência entre itens do backlog, ordenando o backlog na sequência de implementação e entrega (peso 20%);
  - Release plan, contendo a indicação de qual a Sprint que desenvolverá e entregará cada item do backlog, criando um roadmap de entregas, balanceando os pontos de esforço entre Sprints (peso 20%);
  - Detalhamento da Sprint atual do projeto (em curso), com Tarefas detalhadas com suas descrições, pontuações de esforço e dependências técnicas (20%).
- FORMA DE ENTREGA:
  - Entregue o link de acesso ao seu plano de projetos na nuvem.
  - Garanta que o professor esteja cadastrado como membro da organização e administrador do projeto.
  - Entregas fora das especificações implicarão em perda de 10% na nota de avaliação.

# DEVOPS TOOLS & CLOUD COMPUTING 1 | 5

## Objetivo

Nesta Sprint, sua equipe deve implementar **uma solução** baseada em uma das disciplinas a seguir:

- JAVA ADVANCED **OU**
- ADVANCED BUSINESS DEVELOPMENT WITH .NET

A solução deve ser implementada **junto com um banco de dados** na nuvem.

Escolha **uma** das opções abaixo:

### ◆ Opção 1: ACR + ACI

Utilize **Azure Container Registry (ACR)** para armazenar sua imagem Docker e **Azure Container Instance (ACI)** para executar o container. **Ambos** devem ser utilizados.

### ◆ Opção 2: Serviço de Aplicativo (App Service)

Publique sua aplicação em um **App Service na Azure** (modelo PaaS), com o banco de dados também na nuvem.

# DEVOPS TOOLS & CLOUD COMPUTING 2 | 5

## Requisitos Obrigatório

Sua entrega **deve** conter os itens abaixo, independentemente da opção escolhida:

1. **Descrição da Solução:** explique brevemente o que a aplicação faz.
2. **Descrição dos Benefícios para o Negócio:** explique quais problemas a solução resolve ou quais melhorias ela traz.
3. **Banco de Dados em Nuvem (**obrigatório**):**
  - **Não serão aceitos:** H2 e Oracle na nuvem da FIAP.
  - Serão aceitos **Bancos em Containers** na **nuvem** ou Bancos com serviços de **PaaS**. Por exemplo: Oracle (Container em Nuvem, OCI), MySQL, Azure SQL (Azure PaaS), PostgreSQL, MongoDB etc.
4. **Implementar um CRUD completo** (Inclusão, Alteração, Exclusão e Consulta) sobre ao menos **uma tabela** da aplicação.
5. Inserir e manipular **pelo menos 2 registros reais** nessa tabela.
6. **Código-fonte publicado no **GitHub**.**
7. **Arquivo PDF** contendo:
  - **Nome completo** e **RM** de **todos** os integrantes
  - Link do **repositório** no GitHub
  - Link do **vídeo** no YouTube

# DEVOPS TOOLS & CLOUD COMPUTING 3 | 5

## Requisitos Específicos Por Tipo De Entrega

Se escolher **ACR + ACI**:

- 8.1. Use apenas imagens oficiais do Docker Hub ou de provedores confiáveis (Azure, OCI, AWS etc)
- 8.2. O container não pode rodar como **root** ou **admin (com privilégios administrativos)**
- 8.3. Pode usar **Dockerfile** ou **Docker Compose**
- 8.4. Entregue todos os scripts do build e execução da imagem, como: Dockerfile, docker-compose.yml (se for o caso),  
Comandos utilizados: docker build, docker push, docker run etc.

Se escolher **Serviço de Aplicativo (App Service)**:

- 9.1. Todos os recursos (App e Banco de Dados) devem ser criados via **Azure CLI**
- 9.2. Entregue todos os scripts dos recursos criados na Azure, como: Grupo de recurso, Plano do serviço, Serviço de aplicativo, Banco de dados e Configurações adicionais.



# DEVOPS TOOLS & CLOUD COMPUTING 4.1 | 5

## Critérios de Avaliação (Pontuação)

1. Desenho da arquitetura da solução proposta com fluxos, recursos e explicação do funcionamento, baseado na disciplina de *DevOps Tools e Cloud Computing* (até 10 pontos).
2. DDL das tabelas (tabelas, colunas, chave primária, comentários etc) criado em arquivo de texto separado somente com esse DDL com o nome: **`script_bd.sql`** com estrutura e comentários (até 10 pontos).
3. Repositório no GitHub separado (crie um repositório para a entrega da disciplina *DevOps Tools e Cloud Computing*), **com tudo que é necessário para a execução do projeto e README.md explicativo (com passo a passo para realizar o deploy e testes**, incluindo os scripts de testes efetuados (POST/PUT em JSON, se for API) (até 10 pontos).

### 4.1 Vídeo Demonstrativo da Solução – (até 70 pontos):

Gravar um vídeo (mínimo 720p, com **áudio claro** e explicação **por voz, sem uso de legendas nessa entrega**) mostrando todo o funcionamento da solução, incluindo:

- Clone do repositório no GitHub
- Deploy da aplicação seguindo exatamente os passos descritos no README.md
- Criação, configuração e testes do App e do **Banco de Dados na nuvem**

# DEVOPS TOOLS & CLOUD COMPUTING 4.2 | 5

## Critérios de Avaliação (Pontuação)

4.2 Demonstração **detalhada** e **individual** de todas as **operações do CRUD** diretamente no Banco de Dados:

Inserção de um registro → exibir no banco

Atualização do registro → exibir no banco

Exclusão do registro → exibir no banco

Consulta de registros

Evidenciar claramente a **integração total** entre o App e o Banco em nuvem, com tudo **funcionando 100%** ⚠

Observações:

Não há limite de tempo para o vídeo, mas **evite excesso de duração**.

A apresentação deve ser **clara, organizada e completa**, evidenciando **todas as etapas** do processo.

# DEVOPS TOOLS & CLOUD COMPUTING 5 | 5

## Penalidades (Descontos)

- ✗ Sem item 1 (descrição da solução): -10 pontos
- ✗ Sem item 2 (benefício para o negócio): -10 pontos
- ✗ Sem um dos itens obrigatório 3, 4 ou 5: -40 pontos
- ✗ Sem repositório separado da disciplina: -10 pontos
- ✗ Sem código-fonte (item 6): -40 pontos
- ✗ Sem o PDF com nome/RM e links (item 7): Nota Zero
- ✗ Para opção ACR + ACI:
  - 8.1: sem imagem oficial: -15 pontos
  - 8.2: container com privilégio admin: -10 pontos
  - 8.4: script faltando: -10 pontos por script
- ✗ Para opção App Service:
  - 9.1: recursos não criados via CLI: -30 pontos
  - 9.2: script faltando: -10 pontos por script

- ✗ Sem README.md com orientações de deploy/teste (Item 3 da avaliação): -30 pontos
- ✗ Sem evidência clara de cada operação CRUD no vídeo (Item 4 da avaliação): -30 pontos
- ✗ Vídeo com baixa qualidade ou sem explicação falada: -30 pontos

Bom trabalho! ;)



# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

Descrição:

**Avançar na implementação técnica da solução**, demonstrando a integração entre os componentes propostos (IoT e/ou Visão Computacional), apresentando um protótipo funcional, mesmo que parcial, e evidências das métricas de performance quantitativa/qualitativa.

- Requisitos: Escolha uma ou mais abordagens abaixo, conforme seu projeto:
- Caso IoT:
  - Prototipagem (real ou simulada) com pelo menos 3 sensores/atuadores distintos e comunicação em tempo real via MQTT e/ou HTTP.
  - Interface gráfica com dados de telemetria ou status das motos (pode ser dashboard simples).
  - Registro persistente (arquivo ou banco de dados) do histórico dos dados coletados e status dos atuadores.
  - Teste funcional com casos de uso realistas (ex: moto desaparecida, moto colocada no lugar errado, etc.) Demonstrar o sistema em funcionamento com ao menos 3 dispositivos IoT (simuladores executando em paralelo).
- Caso Visão Computacional:
  - Script funcional de rastreamento ou detecção de múltiplas motos com output visual com as detecções destacadas em tempo real com uso de algoritmos como YOLO, MediaPipe, etc.

# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

Pontuação:

Critério	Pontuação
Comunicação entre sensores/visão e backend	até 30 pts
Dashboard/output visual com dados em tempo real	até 30 pts
Persistência e estruturação dos dados	até 20 pts
Organização do código e documentação técnica	até 20 pts

Penalidades, com suas pontuações.

Critério	Desconto
Ausência de vídeo explicativo	-20 pts
Sem dashboard funcional ou output visual	-30 pts
Dados não persistidos e sem integração com backend	-30 pts
Projeto não funcional ou incoerente com desafio	-60 pts

# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

Entregáveis obrigatórios: Arquivo .zip contendo:

- Link do vídeo publicado no YouTube com a apresentação da ideia central do projeto e demonstração funcional (real ou simulada).
- Link para o repositório no GitHub, apresentando código-fonte e estrutura do projeto e o README com instruções de uso, tecnologias utilizadas e resultados parciais.

Condições de entrega

- A integridade e o conteúdo do arquivo entregue são de responsabilidade dos integrantes do grupo.
- Arquivos entregues sem conteúdo ou com arquivos corrompidos não serão considerados. Sugestão: confira seu anexo antes de publicar.
- Não serão aceitos arquivos enviados pelo Teams ou fora do prazo.

# JAVA ADVANCED <sup>1/3</sup>

Você deverá desenvolver uma aplicação web completa utilizando o framework Spring Boot para dar suporte à solução proposta para a Mottu. Esta aplicação deve ter o foco nos seguintes tópicos:

1. Thymeleaf para a camada de visualização (frontend);
2. Flyway para controle de versões do banco de dados;
3. Spring Security para autenticação e controle de acesso.

## Requisitos

A aplicação deverá conter obrigatoriamente os seguintes elementos técnicos:

1. Thymeleaf (30 pontos)
  - Páginas HTML utilizando Thymeleaf para listar, criar, editar e excluir registros.
  - Utilização de fragmentos (ex: cabeçalho, rodapé, menu) para evitar repetição de código.
2. Flyway (20 pontos)
  - Configuração do Flyway para versionamento do banco de dados.
  - Mínimo de quatro versões de migração (por exemplo: criação de tabelas e inserção de dados iniciais).
3. Spring Security (30 pontos)
  - Sistema de autenticação via formulário (login e logout).
  - Implementação de pelo menos dois tipos de usuário, com permissões diferentes.
  - Proteção de rotas com base no perfil do usuário.

# JAVA ADVANCED 2/3

## 4. Funcionalidades completas (20 pontos)

- Implementação funcional de pelo menos dois fluxos completos do sistema (exceto CRUD)
- Validações básicas nos formulários e nos dados.

## Penalidades

As penalidades abaixo serão aplicadas independentemente dos requisitos técnicos, e dizem respeito à qualidade do código e funcionamento geral do sistema:

Problema	Desconto
Violação evidente de princípios SOLID (ex: métodos gigantes, responsabilidades múltiplas)	-10 pontos por ocorrência
Código com repetições desnecessárias (violação de <b>DRY</b> )	-5 pontos por ocorrência
Código com problemas de legibilidade (violação de <b>Clean Code</b> — nomes ruins, ausência de métodos claros, comentários inúteis)	-5 pontos por ocorrência
Comentários no lugar de refatorações claras (explicações de código ruim ao invés de clareza no próprio código)	-3 pontos
Funcionalidade com <b>comportamento inesperado ou erro evidente</b>	-5 pontos por funcionalidade
Página que não carrega ou <b>link quebrado</b>	-5 pontos por página/link
Apresentação incompleta ou incoerente com a proposta do challenge.	-15 pontos

# JAVA ADVANCED <sup>3/3</sup>

## Entrega

Os seguintes artefatos devem ser entregues no portal até o prazo definido. Não serão aceitas entregas após o prazo ou por outros meios.

- Link do repositório público do GitHub com o código fonte da aplicação Spring completa.
- O readme do repositório deve conter todas as instruções necessárias para a instalação, execução e acesso da aplicação.
- Vídeo demonstração da aplicação funcionando com apresentação das principais funcionalidades da aplicação web (máx. 10 min.)

## Avaliação Oral

Após a entrega do projeto, cada aluno participará de uma avaliação oral individual em sala de aula. Essa etapa complementa a avaliação técnica. O objetivo é verificar a compreensão do código entregue e promover o uso consciente de ferramentas como a inteligência artificial. Durante essa conversa, o aluno deverá:

- Explicar trechos específicos do próprio código;
- Justificar decisões de implementação;
- Comentar eventuais dificuldades encontradas;
- Descrever se e como utilizou ferramentas de IA no processo.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- Demanda: Desenvolver **02 procedimentos, 02 funções, 01 gatilho (trigger)** e os scripts completos de **estrutura e carga de dados** utilizando o banco de dados relacional **Oracle**. A atividade também exige a entrega de **documentação técnica com prints de execução e tratamento de exceções** implementado em formato .PDF e .SQL.
- O trabalho tem como foco:
  - Desenvolver lógica procedural SQL estruturada;
  - Manipular dados relacionais e convertê-los para JSON sem uso de funções automáticas;
  - Implementar auditoria via trigger;
  - Aplicar boas práticas de documentação, tratamento de erros e organização de código.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- Especificações:
- Procedimentos (30 pontos)

## A.Procedimento 1:

- Realizar JOIN entre **duas ou mais tabelas relacionais do seu projeto**.
- Exibir os dados no **formato JSON (string)**.
- A transformação relacional para JSON deve ser feita manualmente com uma **função criada pelo grupo (ver Função 1)**.
- Cada tabela usada deve conter **no mínimo 5 registros válidos**.
- Tratar no mínimo **3 exceções distintas** com EXCEPTION WHEN.



# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- Especificações:

## B. Procedimento 2:

- Ler os dados de uma tabela de fatos que contenha, pelo menos, **duas colunas categóricas** (por exemplo, agencia e conta) e uma coluna numérica com valores a serem somados (por exemplo, saldo). Esse procedimento deve calcular e exibir:
  - Os **valores somados por combinação completa** das categorias (agencia, conta);
  - Um **subtotal por grupo da primeira categoria** (agencia);
  - Um **total geral** ao final da listagem.
- As somas devem ser feitas diretamente dentro do procedimento, sem o uso de recursos automáticos do Oracle. Os grupos de subtotal devem estar na mesma coluna e linha da saída, com os valores de agrupamento ausentes (nulos) nos casos de subtotal e total geral.
- A lógica de somatório deve ser **manual**, feita dentro do corpo do procedimento.
- **Não é permitido** o uso de funções automáticas de agregação avançada como ROLLUP, CUBE, GROUPING SETS, GROUPING, ou semelhantes.
- O procedimento deve conter **tratamento de pelo menos 3 exceções distintas**.
- A tabela utilizada deve conter **pelo menos 5 linhas detalhadas** para que haja dados suficientes para os cálculos de subtotal e total geral.
- A ordem de exibição deve seguir a hierarquia das categorias.
- A exibição deve seguir a ordem de agrupamento e apresentar o resultado no seguinte formato:

Agencia	Conta	Saldo
-----	-----	-----
1	1	4363.55
1	2	4794.76
1	3	4718.25
1	4	5387.45
1	5	5027.34
Sub Total		24291.35
2	1	5652.84
2	2	4583.02
2	3	5555.77
2	4	5936.67
2	5	4508.74
Sub Total		26237.04
Total Geral		50528.39

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- Funções (30 Pontos)
  - A. Função 1:
    - Crie uma Função que recebe dados relacionais e retorna uma string no **formato JSON**.
    - A lógica de conversão deve ser desenvolvida pelo grupo.
    - **Proibido o uso de funções internas/built-in do Oracle**, como TO\_JSON, JSON\_OBJECT, JSON\_VALUE, JSON\_QUERY, JSON\_TABLE, ou similares.
    - Tratar no mínimo **3 exceções distintas**.
  - B. Função 2:
    - Crie uma Função que substitui algum processo lógico do projeto, como:
      - Validação de senha, cálculos matemáticos, verificação de limites, etc.
    - Tratar no mínimo **3 exceções distintas**.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- **Trigger (30 pontos)**
- Trigger de Auditoria (DML)
- Criar uma tabela de auditoria com os campos:
  - Nome do usuário
  - Tipo da operação (INSERT, UPDATE, DELETE)
  - Data e hora da operação
  - Valores anteriores (:OLD)
  - Valores novos (:NEW)
- Criar uma trigger AFTER INSERT OR UPDATE OR DELETE ON <tabela> que grave esses dados em cada operação realizada.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

- **Entrega e Documentação (10 pontos)**

- a) Arquivo PDF: 2TDSX\_2024\_Proj\_BD.pdf

- Capa com:
      - **Nomes completos dos integrantes**
      - **RMs**, em ordem alfabética
    - Prints de tela que mostrem:
      - Execução de cada função/procedimento/gatilho
      - Pelo menos **uma exceção tratada** por função e procedimento
    - Código comentado e bem organizado
    - Código da 2ª Sprint corrigido conforme feedback anterior

- b) Arquivo SQL: 2TDSX\_2024\_CodigoSql\_Integrantes.sql

- Deve conter **todo o código necessário para execução do projeto**, incluindo:
    - CREATE TABLE de todas as tabelas utilizadas
    - INSERT INTO com registros (mínimo 5 por tabela)
    - Procedimentos (CREATE OR REPLACE PROCEDURE)
    - Funções (CREATE OR REPLACE FUNCTION)
    - Trigger de auditoria
    - **Comentários** no código indicando a finalidade de cada bloco

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

Infração	Descrição	Desconto
Uso de funções built-in para JSON	Uso de TO_JSON, JSON_OBJECT, JSON_VALUE, etc.	-10 pontos por ocorrência
Falta de tratamento de exceções	Item sem EXCEPTION WHEN ou equivalente	-5 pontos por item
Ausência de prints com exceções	Não apresentar erro tratado em execução	-5 pontos por item
Código sem comentários ou desorganizado	Dificulta leitura e correção	-5 pontos
Tabelas com menos de 5 registros	Não cumpre requisito mínimo	5 pontos por tabela
Trigger incompleta ou não funciona	Não registra auditoria corretamente	-10 pontos
Falta de documentação mínima	Sem capa, prints, ou explicações claras	-5 a -10 pontos
Não entregar o arquivo .sql completo	Ausência de scripts com CREATE, INSERT, etc.	-10 pontos

# MOBILE APPLICATION DEVELOPMENT

Esta é a entrega intermediária do projeto do aplicativo mobile desenvolvido em React Native. A expectativa é que o app já esteja funcional e reflita boa parte das funcionalidades planejadas. O protótipo entregue anteriormente deve ter sido transformado em uma base concreta de desenvolvimento, com código limpo, integração com a API e interface minimamente refinada.

## Requisitos

1. Telas funcionais integradas com API
2. Sistema de Login
3. Estilização com Tema
4. Arquitetura de Código
5. Documentação e Apresentação

# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 1. Telas funcionais integradas com API (40 pontos)

*Critério de avaliação: profundidade e integridade das integrações realizadas com a API.*

- a. Implementar ao menos duas funcionalidades completas utilizando a API da aplicação (Java ou .NET) (10 pontos)
- b. Cada funcionalidade deve incluir as operações completas relacionadas (ex.: um CRUD deve conter Create, Read, Update, Delete). (10 pontos)
- c. Deve haver tratamento completo de formulários: validações, mensagens de erro, feedback ao usuário. (10 pontos)
- d. Indicadores de carregamento devem ser usados em chamadas de rede. (10 pontos)

### 2. Sistema de Login (20 pontos)

*Critério de avaliação: funcionamento completo do fluxo de autenticação, clareza e segurança mínima no processo. Validação dos formulários, tratamentos de erros e indicadores de carregamento devem estar presentes. Pode utilizar Firebase Authentication ou API desenvolvida nas disciplinas de JAVA ou .NET*

- a. Tela de Login (5 pontos)
- b. Tela de Cadastro (10 pontos)
- c. Logout funcional (5 pontos)

# MOBILE APPLICATION DEVELOPMENT

## CrITÉRIOS de Avaliação e pontuação

### 3. Estilização com Tema (15 pontos)

*Critério de avaliação: aderência às guidelines, qualidade estética e coerência visual.*

- a. O app deve suportar modo claro e modo escuro. (5 pontos)
- b. Deve haver personalização visual consistente com o tema do app: cores, fontes, imagens. (3 pontos)
- c. Seguir as guidelines de design da Google (Material Design) ou Apple (Human Interface Guidelines). (2 pontos)
- d. Demonstrar criatividade e identidade visual coerente. (5 pontos)

### 4. Arquitetura de Código (15 pontos)

- Organização lógica de arquivos, pastas e componentes.
- Nomeação clara e padronizada de variáveis, funções e componentes.
- Separação adequada de responsabilidades (componentes, serviços, estilos, rotas, contextos, etc.).
- Código limpo, legível e bem estruturado.
- Indentação correta e formatação padronizada, com uso consistente de espaçamentos, quebras de linha e nomes descritivos.
- Uso de boas práticas específicas do desenvolvimento com React Native.
- Utilização de ferramentas como ESLint, Prettier ou o formatador do editor, conforme configurado no projeto.
- As bibliotecas utilizadas devem ser relevantes, atualizadas e necessárias ao escopo. Pode haver penalização por uso excessivo, desatualizado ou não justificado.



# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 5. Documentação e Apresentação (10 pontos)

- O repositório deve conter um arquivo README.md com:
  - Nome do app
  - Proposta e funcionalidades
  - Estrutura de pastas
  - Nome, RM e GitHub de todos os integrantes
- Gravação de vídeo demonstrando o app em funcionamento real (emulador ou dispositivo), apresentando todas as funcionalidades.

# MOBILE APPLICATION DEVELOPMENT

## Penalidades

- I. Não entregar via GitHub Classroom (-20 pontos)
- II. Não entregar vídeo de apresentação (-20 pontos)
- III. Ausência do arquivo README.md (-10 pontos)
- IV. Não utilizar tema (modo claro/escuro) (-20 pontos)
- V. Aplicativo fora do escopo das aulas (-60 pontos)
- VI. Remoção de telas entregue na Sprint anterior (-100 pontos)
- VII. Histórico do Git incoerente ou confuso (-50 pontos)

É esperado que o repositório tenha uma árvore de commits sequencial e evolutiva, com mensagens claras e representando a construção real do app. Repositórios que não demonstrem envolvimento prático ou com uso superficial do Git poderão ser penalizados.

# 4º ENTREGAS

POR DISCIPLINA

# ADVANCED BUSINESS DEVELOPMENT WITH .NET

- Finalizar a API RESTful utilizando .NET (Web API ou Minimal API) com foco em boas práticas REST
  - 10 pts – Implementar um endpoint de Health Checks
  - 10 pts - Implementar versionamento da API
  - 25 pts – Implementar segurança de API (API KEY ou JWT)
  - 25 pts – Implementar ao menos um endpoint que use o ML.NET
- 30 pts - Testes unitários com xUnit para lógica principal
  - Testes de integração básicos com WebApplicationFactory
  - README deve conter instruções para executar os testes

# ADVANCED BUSINESS DEVELOPMENT WITH .NET

- Penalidades
  - -20pts – Não atualizar a documentação do Swagger
  - -100pts – Projeto não compilar
  - -20pts – Não atualizar a documentação do Readme no Github

# COMPLIANCE, QUALITY ASSURANCE & TESTS

- SPRINT 4 - COMPLIANCE, QUALITY ASSURANCE & TESTS
- ITENS OBRIGATÓRIOS:
  - PARTE A) Crie o plano de testes manuais, de validação no nível de sistema para seu projeto (as funcionalidades principais do seu sistema precisam estar cobertas), usando a plataforma AZURE BOARDS:
    - 1) Liste os testes planejados – peso 20%
    - 2) Descreva os dados de entrada para cada teste – peso 20%
    - 3) Descreva os dados de saída para cada teste – peso 20%
    - 4) Descreva o procedimento (passos) de teste aplicado – peso 20%
    - Obs: trabalhe com dados controlados nos seus testes, predefinindo os valores de variáveis para input e output esperado. Os testes precisam estar de acordo com o plano de release e tarefas de sprints realizadas. Esses itens serão checados e em caso de desrespeito às regras, será aplicada uma penalidade de 10% na nota.
  - PARTE B) Utilize um software de automação para validar o seu sistema. Pode ser usada automação Record & Playback com Selenium IDE ou Katalon Studio se o seu sistema tiver telas de interface de usuário. Se o seu sistema é composto somente por APIs e componentes de serviços sem interface de usuário, empregue POSTMAN ou software equivalente na automação:
    - 5) Faça ao menos 4 casos de testes automatizados para a sua aplicação – peso (20%)
- FORMA DE ENTREGA:
  - Crie um repositório no GIT HUB, com acesso público e suba na Branch develop: Link de acesso ao AZURE BOARDS para correção dos testes manuais, sendo que o professor precisa ser membro da organização e do projeto para poder acessar e corrigir o trabalho; Link de vídeo que mostra a configuração e execução dos testes de automação, nas ferramentas escolhidas.
  - Entregas fora do formato implicarão em penalidade de 10% na nota.

# DEVOPS TOOLS & CLOUD COMPUTING 1 | 4.2

## Objetivo e Entrega Obrigatória

Nesta última Sprint, sua equipe deve implementar **uma solução** baseada em uma das disciplinas a seguir:

- JAVA ADVANCED OU
- ADVANCED BUSINESS DEVELOPMENT WITH .NET

juntamente com um **Banco de dados em nuvem**. Onde deverá utilizar a ferramenta **Azure DevOps** e os recursos da **Azure** para criar e configurar **pipelines** necessárias para as práticas de **CI/CD**.

### Entrega Obrigatória

Um **PDF** com:

- Nome completo, RM dos integrantes do grupo e **turma** de **cada** integrante
- Link do repositório no GitHub
- Link do vídeo no YouTube
- Link do projeto no Azure DevOps
- Itens 1, 2 e 3 dos requisitos (próximo slide)

**Links quebrados ou sem acesso = trabalho não entregue (nota zero)**



# DEVOPS TOOLS & CLOUD COMPUTING 2.1 | 4.2

## Requisitos

### 1. Descrição da solução

Explique brevemente o que sua aplicação faz e informe a stack de tecnologias utilizadas.

### 2. Diagrama da Arquitetura + Fluxo CI/CD

- Inclua todos os componentes da Stack;
- Mostre o fluxo das esteiras CI/CD com números indicando a ordem;
- Adicione as personas (usuário final, desenvolvedor etc.);
- Utilize setas/linhas para mostrar conexões;
- Indique onde cada ferramenta atua;

### 3. Detalhamento dos componentes. Exemplo:

Nome do componente	Tipo	Descrição funcional	Tecnologia/Ferramenta
Repositório de código	SCM	Onde o código-fonte está versionado	GitHub
Pipeline	Orquestrador CI	Compila e executa testes unitários	Azure DevOps Pipelines





# DEVOPS TOOLS & CLOUD COMPUTING 2.2 | 4.2

## Requisitos

### 4. Banco de Dados em Nuvem (obrigatório)

**Não serão aceitos:** H2, MongoDB Atlas e Oracle na nuvem da FIAP.

**Aceitos:** Bancos em Containers na nuvem ou Bancos com serviços de PaaS. Exemplo: MySQL, PostgreSQL, Azure SQL (PaaS), MongoDB, Oracle (Container em Nuvem, OCI) etc.

### 5. Configuração do Projeto no Azure DevOps com as seguintes informações:

**Project Name:** Sprint 4 – Azure DevOps

**Description:**

- Projeto para entrega da Sprint 4 do professor <nome do professor>
- Integrantes do grupo. Exemplo: <RM00000> - <nome do aluno> - <turma 2TDS..>

**Visibility:** Private

**Version control:** Git

**Work item process:** Scrum

### 6. Acesso ao Professor

Convide o professor para o projeto e dê o acesso no nível “Basic”.



# DEVOPS TOOLS & CLOUD COMPUTING 2.3 | 4.2

## Requisitos

### 7. Pipelines CI/CD

Configure no modo **Clássico** ou **YAML** para executar:

- a) *CI*: build + testes automáticos
- b) *CD*: deploy automático

#### Regras da Pipeline (obrigatório):

- I. Deve estar configurada e conectada ao repositório GitHub da aplicação;
- II. O *CI* deve estar configurada para disparar a cada alteração na branch master;
- III. O *CD* disparar após novo artefato gerado;
- IV. Variáveis de ambiente protegidas para dados relacionados à conexão de banco de dados, usuários e senhas (credenciais, conexões);
- V. Para a prática de CI é necessária a geração e publicação do artefato no ambiente do Azure DevOps;
- VI. Para a prática de CI é necessária a etapa de execução de testes;
- VII. Para a prática de CD a aplicação deverá ser provisionada (Deploy) em um dos recursos: **Azure Web App** ou **Azure Container Instance (ACI)**.

⚠ *Independente de qual tecnologia escolhida para provisionar a aplicação, deve-se usar imagem Docker para provisionar a aplicação em um dos recursos escolhidos.*



# DEVOPS TOOLS & CLOUD COMPUTING 2.4 | 4.2

## Requisitos

### 8. Vídeo Demonstrativo

Qualidade mínima 720p, áudio claro, **sem legendas**, explicação por voz;

Ferramentas visíveis e preparadas antes de gravar;

Sem limite de tempo (mas **sem exageros**);

**Dicas:** Prepare o ambiente antes da gravação:

- Deixe as ferramentas de desenvolvimento com o código da sua aplicação **abertas** (IDE, Azure DevOps, Pipelines, Banco de Dados, Portal do Azure etc.), prontas para o uso.
- Certifique-se que é possível realizar a operação de push para o repositório.
- Certifique-se que o som e a imagem esteja bem configurado atendendo os requisitos.
- Não se esqueça de verificar a gravação antes de enviar o vídeo, garantindo que a visualização das ferramentas estejam bem visíveis.
- Se atente que não será aceito legenda nessa entrega.
- É obrigatório a explicação pelo aluno sobre os itens a serem entregues.



# DEVOPS TOOLS & CLOUD COMPUTING 2.5 | 4.2

## Requisitos

### Fluxo do vídeo:

- Mostre as ferramentas usadas (IDE, Azure DevOps, Banco de Dados, Portal do Azure etc.)
- Apresente a configuração da pipeline (Como cada jobs/tasks como solicitado no **Requisito 7**).
- Altere o README.md no projeto e fazer push no GitHub - Na ferramenta de desenvolvimento, com o seu projeto da aplicação em JAVA ADVANCED ou ADVANCED BUSINESS DEVELOPMENT WITH .NET aberto, faça uma alteração no arquivo README.md e realize o push das alterações para o seu repositório no GitHub.
- Mostre o início automático da pipeline após o push.
- Explique **cada** etapa do CI/CD durante a **execução** - Abra os detalhes do fluxo de execução da sua pipeline e comente no vídeo.
- Mostre o artefato criado e os resultados dos testes.
- Demonstre no Portal do Azure os recursos atualizados pelo **deploy** (Web App ou Azure Container Instance).
- Acesse a aplicação no **Web App** ou **ACI** e execute o **CRUD completo**, mostrando no banco **cada** operação (inserir, consultar, atualizar, deletar).
- O CRUD pode ser feito via interface da aplicação ou via chamadas REST, utilizando a ferramenta que desejar.



# DEVOPS TOOLS & CLOUD COMPUTING 3 | 4.2

## Critérios de Avaliação (Pontuação)

### Requisito

- 0 – PDF com links e dados
- 1 – Descrição da solução
- 2 – Diagrama
- 3 – Detalhamento dos componentes
- 4 – Banco de Dados válido
- 5 – Configuração do projeto no Azure DevOps
- 6 – Convite ao professor
- 7 – Pipelines CI/CD funcionando
- 8 – Vídeo completo e claro

### Pontos

**Obrigatório**

5

10

10

**Obrigatório**

**Obrigatório**

**Obrigatório**

30

45

Na falta dos itens considerado **Obrigatório** a avaliação será **zerada!** Sem possibilidade de avaliação.



# DEVOPS TOOLS & CLOUD COMPUTING 4.1 | 4.2

## Penalidades (Descontos)

**IMPORTANTE:** A falta de **qualidade** na entrega dos itens, é passível de desconto.

### Requisito

- 0 – PDF com links e dados
- 1 – Descrição da solução
- 2 – Diagrama
- 3 – Detalhamento dos componentes
- 4 – Banco de Dados válido
- 5 – Configuração do projeto no Azure DevOps
- 6 – Convite ao professor
- 7 – Pipelines CI/CD funcionando
  - 7 - a) Build e testes da aplicação. (CI)
  - 7 - b) Deploy da aplicação. (CD)
  - 7 – I) A pipeline deve estar configurada para o repositório GitHub da sua aplicação
  - 7 – II) A pipeline(CI) deve estar configurada para disparar sempre que houver uma mudança de código na branch master e disparar a pipeline (CD) quando um novo artefato gerado.
  - 7 – III) A pipeline deve conter variáveis de ambiente protegidas para dados relacionados à conexão de banco de dados, usuários e senhas.
  - 7 – IV) Para a prática de CI é necessária a geração e publicação do artefato no ambiente do Azure DevOps (Testes).

### Pontos

A falta de informação do link do vídeo do Youtube ou o link do projeto no GitHub ou o link do projeto no Azure DevOps, a avaliação será Zerada. Não é possível avaliar. Links quebrados ou sem acesso serão considerados como não entregue.

Na falta deste item será descontado - 5 pontos.

Na falta deste item será descontado - 10 pontos.

Na falta deste item será descontado -10 pontos.

**Obrigatório** - Na falta deste item a avaliação será zerada. Não é possível avaliar.

**Obrigatório** - Na falta deste item a avaliação será zerada. Não é possível avaliar.

**Obrigatório** - Se o professor não for convidado para o projeto e não tiver o acesso Basic não irá conseguir acessar o projeto ou a pipeline. A avaliação será zerada. Não é possível avaliar.

Na falta deste item a **avaliação será zerada**. Não é possível avaliar.

Na falta deste item será descontado – 15 pontos.

Na falta deste item será descontado – 15 pontos.

Na falta deste item será descontado - 5 Pontos.

Na falta deste item será descontado - 10 Pontos.

Na falta deste item será descontado – 15 pontos.

Na falta deste item será descontado - 5 pontos.



# DEVOPS TOOLS & CLOUD COMPUTING 4.2 | 4.2

## Penalidades (Descontos)

**IMPORTANTE:** A falta de **qualidade** na entrega dos itens, é passível de desconto.

### Requisito

7 – V) Para a prática de CI é necessária a etapa de testes.

7 – VI) Para a prática de CD a aplicação deverá ser provisionada (Deploy) em um dos recursos: Azure Web App ou Azure Container Instance (ACI).

7 – VII) Independente de qual tecnologia escolhida para provisionar a aplicação, deve se usar imagem Docker para provisionar a aplicação em um dos recursos escolhidos (Azure Web App ou Azure Container Instance (ACI)).

8 – Vídeo completo e claro.

8.1 – Mostre as ferramentas usadas .

8.2 – Apresente a configuração da pipeline.

8.3 – Altere o README.md no projeto e fazer push no GitHub.

8.4 – Mostre o início automático da pipeline após o push.

8.5 - Explique **cada** etapa do CI/CD durante a **execução**.

8.6 – Mostre o artefato criado e os resultados dos testes.

8.7 – Demonstre no Portal do Azure os recursos atualizados pelo deploy

8.8 - Acesse a aplicação a partir do recurso utilizado (Web App ou Azure Container Instance) seja via FQDN configurado ou IP público e realize um CRUD completo. O CRUD pode ser feito via interface da aplicação ou via chamadas REST, utilizando a ferramenta que desejar. Para cada operação do CRUD demonstre os dados no banco de dados.

### Pontos

Na falta deste item será descontado - 5 pontos.

Na falta deste item a **avaliação será zerada**. Não é possível avaliar.

Na falta deste item será descontado – 20 pontos.

**Obrigatório** - Na falta deste item a avaliação será zerada. Não é possível avaliar.

Na falta deste item será descontado - 5 pontos.

Na falta deste item será descontado - 15 pontos.

Na falta deste item será descontado – 10 pontos.

Na falta deste item será descontado - 10 Pontos.

Na falta deste item será descontado - 15 pontos.

Na falta deste item será descontado - 5 pontos.

Na falta deste item será descontado - 10 Pontos.

A falta deste item a avaliação será zerada. Caso exista a demonstração parcial do CRUD o desconto será de - 15 pontos.

# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

- Itens obrigatórios:

- Descrição;

Entrega final da solução funcional, integrada e inovadora, alinhada ao desafio proposto pela Mottu. A entrega deve demonstrar domínio técnico, integração entre disciplinas, clareza na proposta e capacidade de comunicação profissional.

- Requisitos;

- Fluxo completo de dados desde captura (IoT ou visão computacional) até visualização final
  - Dashboard ou Interface Final com usabilidade exibindo:
    - Localização das motos no pátio (mapa, grid ou visão geral)
    - Estado de cada moto (em uso, parada, manutenção, etc.)
    - Alertas ou indicadores em tempo real
  - Integração com outras disciplinas como: Mobile App, Java, .NET, Banco de Dados, DevOps.



# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

Pontuação:

Critério	Pontuação
Funcionalidade técnica da solução (ponta a ponta)	até 60 pts
Integração com demais disciplinas (App, API, Banco, DevOps)	até 20 pts
Apresentação em vídeo (clareza, domínio, coesão)	até 10 pts
Organização do repositório e documentação	até 10 pts

Penalidades, com suas pontuações.

Critério	Desconto
Ausência de vídeo explicativo com todos os membros	-20 pts
Código inconsistente com o vídeo apresentado	-30 pts
Projeto sem conexão clara com o desafio da Mottu	-60 pts
Interface inoperável ou dados não fluindo corretamente	-40 pts

# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

Entregáveis obrigatórios: Arquivo .zip contendo:

- Link do vídeo publicado no YouTube com a apresentação e demonstração funcional (real ou simulada) FINAL.
- Link para o repositório no GitHub, apresentando código-fonte e estrutura do projeto e o README com instruções de uso, tecnologias utilizadas e resultados FINAIS.

Condições de entrega

- A integridade e o conteúdo do arquivo entregue são de responsabilidade dos integrantes do grupo.
- Arquivos entregues sem conteúdo ou com arquivos corrompidos não serão considerados. Sugestão: confira seu anexo antes de publicar.
- Não serão aceitos arquivos enviados pelo Teams ou fora do prazo.

# DISRUPTIVE ARCHITECTURES: IOT, IOB & GENERATIVE IA

- Itens obrigatórios:

- Descrição;

Entrega final do projeto com foco em solução integrada, realista, escalável e inovadora. Esta entrega será avaliada em sua maturidade técnica e capacidade de impacto real. É o momento de conectar todas as disciplinas e demonstrar a viabilidade de implantação.

- Requisitos;
  - Pontuação;
  - Penalidades, com suas pontuações.

# JAVA ADVANCED 1/4

Durante o semestre, seu grupo trabalhou em um cujo desafio central foi propor e implementar uma solução tecnológica inovadora para um problema real da Mottu. Agora é o momento de consolidar todo esse trabalho e apresentar o resultado final de forma clara, funcional e integrada, demonstrando:

- O funcionamento do sistema;
- A coerência da solução proposta com o desafio;
- O uso das tecnologias aprendidas;
- E a conexão com as demais disciplinas do semestre.

## Requisitos da Entrega Final

### 1. Demonstração Técnica da Solução (40 pontos)

- A aplicação deve estar rodando online (deploy)
- A equipe deve navegar pelos principais fluxos do sistema.
- A solução deve aplicar os principais conceitos da disciplina de forma contextualizada com o app criado.
- A interface deve apresentar boa UI e UX.

# JAVA ADVANCED 2/4

## 2. Narrativa da Solução (20 pontos)

- Explicação clara da proposta da solução do grupo.
- Apresentação das decisões de design, escolhas tecnológicas e justificativas.
- Destaque para a originalidade e criatividade da solução.

## 3. Integração Multidisciplinar (20 pontos)

- Explicitação e demonstração de como as demais disciplinas foram aplicadas na solução.
- Apresentação de evidências: documentação, canvas, protótipos, scripts SQL, etc.

## 4. Apresentação Oral e Comunicação em Equipe (10 pontos)

- Todos os membros devem participar do vídeo.
- Clareza, objetividade e domínio sobre o que está sendo demonstrado.

## 5. Organização da entrega e da documentação (10 pontos)

# JAVA ADVANCED 3/4

## Penalidades

Problema	Desconto
Código com violações evidentes de princípios de boas práticas (ex: SOLID, DRY, Clean Code)	-10 pontos por ocorrência
Repetição de código que poderia ter sido extraído para métodos reutilizáveis ou templates	-5 pontos por ocorrência
Ausência de evidência de colaboração entre membros (ex: código concentrado em uma única pessoa, ausência em reuniões)	-10 pontos
Falhas graves de usabilidade na interface	-5 pontos por ocorrência
Erros visuais ou de fluxo lógico durante a apresentação	-5 pontos
Falta de alinhamento com o problema da Mottu (ex: solução genérica que ignora o desafio proposto)	-10 pontos
Entrega fora do prazo ou fora do portal	-100 pontos

# JAVA ADVANCED 4/4

## Artefatos de Entrega

- Repositório no GitHub com README completo.
- Link de acesso à aplicação.
- Vídeo da apresentação com duração máxima de 15 minutos.

## Avaliação Oral Individual (opcional)

Além da apresentação em grupo, poderá ser realizada uma avaliação oral individual com cada integrante, com o objetivo de verificar o domínio técnico sobre o que foi entregue.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

## Itens obrigatórios:

- De acordo com o projeto desenvolvido na Sprint anterior, reentregar a modelagem relacional com as orientações e correções do professor.
- Atualizar a codificação com a inclusão do empacotamento dos processos para uma melhor organização e dinamismo da sua execução. Os Objetos do banco de dados (funções, procedimentos e gatilhos) devem estar empacotados a fim de garantir a modularidade e reutilização do código.
- O empacotamento deve seguir as boas práticas, agrupando os objetos de maneira lógica e acessível.
- Integração com Outras linguagens do Curso (Java, C, Mobile)
- A mesma base de dados relacional deve ser usada para o backend da aplicação, implementada seja em Java, C#, ou uma plataforma mobile.
- As procedures criadas deverão ser chamadas via aplicação e demonstrar a execução no video.



# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

## Importação para MongoDB

- O dataset exportado em JSON deve ser importado para o MongoDB, criando uma estrutura de dados que obedeça à arquitetura de um banco não relacional de documentos.
- A estrutura MongoDB deve ser coerente com os princípios de banco de dados NoSQL, organizando os dados de maneira eficiente e flexível para consultas.

## Demonstração em Vídeo

- Gravar um vídeo demonstrativo mostrando:
- A execução das procedures sendo utilizadas no backend.
- A inserção dos dados no banco relacional.
- A exportação do dataset para JSON e sua importação no MongoDB.

# MASTERING RELATIONAL AND NON-RELATIONAL DATABASE

## Entregáveis:

1. Modelos Lógico e Físico do banco de dados relacional (formato pdf), codificação (formato sql) Sprint 3 - 20 pontos
2. Empacotamento dos procedimentos, funções e gatilhos personalizadas com tratamento de exceções (em sql). - 25 pontos
3. Arquivo JSON gerado a partir do banco relacional. - 20 pontos
4. Estrutura MongoDB (código fonte para criação e importação dos dados em MongoDB). - 25 pontos
5. Vídeo demonstrativo da execução de todas as partes descritas – 10 pontos

## 6. Penalidades:

Arquivos fora do padrão, menos 10 pontos.

Video sem narração, menos 5 pontos

Execução de processos com erros, sem avaliação

# MOBILE APPLICATION DEVELOPMENT

Esta é a entrega final. Vocês devem apresentar a versão completa e funcional do aplicativo desenvolvido em React Native. Espera-se que a entrega represente um produto pronto para uso, com todas as funcionalidades planejadas implementadas, publicado e documentado. Este é o momento de demonstrar maturidade técnica, atenção aos detalhes e a capacidade de transformar uma ideia em uma solução mobile de verdade.

## Requisitos

1. Implementação funcional de todas as telas do app
2. Publicação do app
3. Notificação via Push
4. Integração com API
5. Localização e Internacionalização
6. Estilização com Tema
7. Arquitetura de Código (React Native)
8. Documentação e Apresentação

# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 1. Implementação funcional de todas as telas do app (30 pontos)

*Critério de avaliação: completude funcional e ausência de bugs.*

- a. Todas as telas planejadas devem estar presentes e 100% funcionais.
- b. A navegação deve estar integrada e fluida.
- c. Deve haver tratamento completo de formulários: validações, mensagens de erro, feedback ao usuário.
- d. Indicadores de carregamento devem ser usados em chamadas de rede.
- e. Todos os botões, interações e chamadas de API devem estar operacionais.

# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 2. Publicação do app (10 pontos)

*Critério de avaliação: acesso funcional à versão publicada e correspondência com o código entregue..*

- a. O aplicativo deve estar publicado no Firebase App Distribution.
- b. O e-mail do professor deve ser adicionado como tester.
- a. A versão publicada deve corresponder exatamente ao código-fonte enviado.
- b. Criar uma tela "Sobre o App", que exiba o hash do commit de referência.

### 3. Notificação via Push (10 pontos)

*Critério de avaliação: troca automática de idioma e consistência das traduções.*

- a. O aplicativo deve implementar ao menos um cenário realista de envio e recepção de notificações push.
- b. A funcionalidade deve ser testável e demonstrável no vídeo.
- c. A escolha do tipo de notificação fica livre, desde que tenha relevância para o contexto da aplicação (ex: nova moto, lembrete, atualização, etc.).

# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 4. Integração com API (10 pontos)

*Critério de avaliação: profundidade e integridade das integrações realizadas com a API.*

- a. Implementar ao menos duas funcionalidades completas utilizando a API da aplicação (Java ou .NET) (3 pontos)
- b. Cada funcionalidade deve incluir as operações completas relacionadas (ex.: um CRUD deve conter Create, Read, Update, Delete). (3 pontos)
- c. Deve haver tratamento completo de formulários: validações, mensagens de erro, feedback ao usuário. (2 pontos)
- d. Indicadores de carregamento devem ser usados em chamadas de rede. (2 pontos)

### 5. Localização e Internacionalização (10 pontos)

*Critério de avaliação: troca automática de idioma e consistência das traduções.*

- a. O app deve suportar os idiomas Português e Espanhol. (5 pontos)
- b. Todas as strings visíveis ao usuário devem estar traduzidas e gerenciadas por arquivos de recursos. (5 pontos)

# MOBILE APPLICATION DEVELOPMENT

## CrITÉRIOS de Avaliação e pontuação

### 6. Estilização com Tema (10 pontos)

*Critério de avaliação: aderência às guidelines, qualidade estética e coerência visual.*

- a. O app deve suportar modo claro e modo escuro. (2 pontos)
- b. Deve haver personalização visual consistente com o tema do app: cores, fontes, imagens. (2 pontos)
- c. Seguir as guidelines de design da Google (Material Design) ou Apple (Human Interface Guidelines). (2 pontos)
- d. Demonstrar criatividade e identidade visual coerente. (4 pontos)

### 7. Arquitetura de Código (10 pontos)

- Organização lógica de arquivos, pastas e componentes.
- Nomeação clara e padronizada de variáveis, funções e componentes.
- Separação adequada de responsabilidades (componentes, serviços, estilos, rotas, contextos, etc.).
- Código limpo, legível e bem estruturado.
- Indentação correta e formatação padronizada, com uso consistente de espaçamentos, quebras de linha e nomes descritivos.
- Uso de boas práticas específicas do desenvolvimento com React Native.
- Utilização de ferramentas como ESLint, Prettier ou o formatador do editor, conforme configurado no projeto.
- As bibliotecas utilizadas devem ser relevantes, atualizadas e necessárias ao escopo. Pode haver penalização por uso excessivo, desatualizado ou não justificado.

# MOBILE APPLICATION DEVELOPMENT

## Critérios de Avaliação e pontuação

### 8. Documentação e Apresentação (10 pontos)

- O repositório deve conter um arquivo README.md com:
  - Nome do app
  - Proposta e funcionalidades
  - Estrutura de pastas
  - Nome, RM e GitHub de todos os integrantes
- Gravação de vídeo demonstrando o app em funcionamento real (emulador ou dispositivo), apresentando todas as funcionalidades.



# MOBILE APPLICATION DEVELOPMENT

## Penalidades

- I. Não entregar via GitHub Classroom (-20 pontos)
- II. Não entregar vídeo de apresentação (-20 pontos)
- III. Ausência do arquivo README.md (-10 pontos)
- IV. Não publicar o app (-40 pontos)
- V. Não utilizar tema (modo claro/escuro) (-20 pontos)
- VI. Aplicativo fora do escopo das aulas (-60 pontos)
- VII. Remoção de telas entregue na Sprint anterior (-100 pontos)
- VIII. Histórico do Git incoerente ou confuso (-50 pontos)

É esperado que o repositório tenha uma árvore de commits sequencial e evolutiva, com mensagens claras e representando a construção real do app. Repositórios que não demonstrem envolvimento prático ou com uso superficial do Git poderão ser penalizados.

