**1. Introdução ao Projeto**

Este documento apresenta a Plataforma de Análise de Endividamento, um sistema desenvolvido para auxiliar indivíduos na compreensão e gestão de seu nível de endividamento financeiro. A plataforma foi concebida com o objetivo primordial de oferecer uma ferramenta que, a partir de dados financeiros fornecidos pelo usuário, calcule o percentual de endividamento e gere recomendações personalizadas, visando fomentar a conscientização financeira e apoiar decisões estratégicas para a saúde econômica.

A aplicação foi desenvolvida utilizando a linguagem de programação Java, na versão 17, e o framework Spring Boot, reconhecido por sua capacidade de agilizar o desenvolvimento de aplicações robustas. Para a persistência de dados, optou-se pelo sistema de gerenciamento de banco de dados MySQL, configurado através do HeidiSQL, e a interação com o banco de dados é realizada por meio do Spring Data JPA e Hibernate. A gestão de dependências e o ciclo de vida do projeto são controlados pelo Apache Maven 3.9.9. O ambiente de desenvolvimento integrado utilizado foi o Spring Tools Suite (STS), baseado no Eclipse.

Este projeto foi desenvolvido no âmbito da disciplina de **Sistemas Distribuídos e Mobile**.

**2. Participantes do Projeto e Atribuições**

O desenvolvimento deste projeto contou com a colaboração das seguintes alunas:

* **Fernanda Pereira da Silva** - RA: 172317393
* **Sara Ferreira de Lara** - RA: 172216038
* **Kauane Demenjon Meira** - RA: 172312644
* **Brenda Sthefany Demenjon Meira** - RA: 172312641

As responsabilidades foram distribuídas da seguinte forma: as alunas Kauane, Brenda e Sara foram as principais responsáveis pela concepção e implementação do código-fonte do sistema. A aluna Fernanda dedicou-se às etapas de testes e à elaboração da presente documentação do projeto.

**3. Funcionalidades da Plataforma**

A Plataforma de Análise de Endividamento oferece um conjunto de funcionalidades CRUD (Create, Read, Update, Delete) para o gerenciamento de registros de usuários financeiros, complementadas por uma lógica de negócio específica para a análise de endividamento. As operações disponíveis são:

* **Cadastro (Create):** Permite a inclusão de novos registros de usuários financeiros, englobando informações essenciais como nome, renda mensal, despesas fixas e o montante total de dívidas.
* **Consulta (Read):** A plataforma possibilita a recuperação de dados de duas formas:
  + Listagem completa de todos os usuários financeiros cadastrados no sistema.
  + Busca individual de um usuário por meio de seu identificador único (ID), fornecendo os dados detalhados do usuário juntamente com uma análise e recomendação sobre seu nível de endividamento.
* **Atualização (Update):** Garante a capacidade de modificar as informações financeiras de um usuário já existente, assegurando a atualidade dos dados.
* **Exclusão (Delete):** Permite a remoção de registros de usuários financeiros do sistema.
* **Cálculo e Recomendação de Endividamento:** Esta é a funcionalidade central do sistema. Com base na renda mensal e no total de dívidas de um usuário, a aplicação calcula o percentual de endividamento. Em seguida, classifica esse percentual em diferentes categorias de risco (e.g., "Situação controlada", "Atenção: alto nível de endividamento", "Crítico: comprometimento total da renda", "Emergência Financeira!") e oferece sugestões de pagamento e estratégias financeiras apropriadas para cada nível de endividamento.

**4. Arquitetura e Estrutura do Código**

O projeto adota uma arquitetura em camadas, um padrão consolidado em aplicações Spring Boot, que promove a separação de responsabilidades e facilita a manutenção e escalabilidade do sistema. A estrutura de diretórios, conforme observado na imagem fornecida, reflete essa organização:

* **com.unicuritiba.plataforma\_endividamento (Pacote Raiz):** Este pacote abriga a classe principal da aplicação, PlataformaEndividamentoApplication.java, responsável pela inicialização do ambiente Spring Boot.
* **controller:** A camada de controladores (Controllers) é responsável por expor os *endpoints* da API REST. As classes nesta camada (UsuarioFinanceiroController.java) recebem as requisições HTTP do cliente, orquestram a chamada aos serviços de negócio apropriados e formatam as respostas HTTP. Inclui também uma classe interna, UsuarioComRecomendacao, para estruturar respostas personalizadas.
* **model:** Esta camada define o modelo de dados da aplicação.
  + UsuarioFinanceiro.java: Representa a entidade principal do sistema, mapeada diretamente para uma tabela no banco de dados. Contém os atributos que descrevem o usuário e suas informações financeiras (id, nome, rendaMensal, despesasFixas, totalDividas).
  + repository (subpacote de model): Contém as interfaces de repositório, como UsuarioFinanceiroRepository.java. Estas interfaces estendem JpaRepository do Spring Data JPA, fornecendo automaticamente os métodos essenciais para operações CRUD com o banco de dados sem a necessidade de implementação manual de SQL.
* **service:** A camada de serviços (Services) encapsula a lógica de negócio da aplicação. As classes de serviço (UsuarioFinanceiroService.java) interagem com os repositórios para manipular os dados e implementam as regras de negócio complexas, como o cálculo do nível de endividamento e a geração de recomendações.

**5. Configuração e Procedimentos de Instalação**

Para que a Plataforma de Análise de Endividamento possa ser configurada e executada em um ambiente local, os seguintes pré-requisitos e passos devem ser observados:

* **Pré-requisitos de Software:**
  + Java Development Kit (JDK) versão 17 ou superior.
  + Apache Maven, versão 3.9.9 ou superior.
  + Servidor MySQL instalado e em pleno funcionamento.
  + HeidiSQL, ou qualquer outro cliente MySQL, para gerenciamento do banco de dados.
  + Spring Tools Suite (STS) para Eclipse, ou uma IDE Java alternativa como IntelliJ IDEA ou VS Code.
* **Passos para Execução:**
  + **Configuração do Banco de Dados MySQL:** É necessário criar o esquema do banco de dados. Através do HeidiSQL, conecte-se ao seu servidor MySQL e execute o comando SQL CREATE DATABASE plataforma\_endividamento;. As tabelas correspondentes às entidades Java serão criadas automaticamente pelo Hibernate na primeira execução da aplicação, devido à configuração spring.jpa.hibernate.ddl-auto=update no arquivo application.properties.
  + **Ajuste das Credenciais de Conexão:** No arquivo src/main/resources/application.properties, as credenciais de acesso ao banco de dados MySQL devem ser verificadas e, se necessário, ajustadas para corresponder à sua instalação local. As configurações atuais são:

Properties

spring.application.name=plataforma-endividamento

spring.datasource.url=jdbc:mysql://localhost:3305/plataforma\_endividamento?useSSL=false&serverTimezone=UTC

spring.datasource.username=admin

spring.datasource.password=dix1bolt

É crucial verificar a porta do MySQL (padrão é 3306, mas no projeto está 3305), o usuário e a senha.

* + **Importação do Projeto na IDE:** Dentro do Spring Tools Suite, o projeto deve ser importado acessando File -> Import... -> Maven -> Existing Maven Projects. Navegue até o diretório raiz do projeto (plataforma\_endividamento) e confirme a importação.
  + **Execução da Aplicação:** No Project Explorer do Eclipse/STS, localize a classe PlataformaEndividamentoApplication.java no pacote src/main/java/com.unicuritiba.plataforma\_endividamento. Clique com o botão direito do mouse sobre esta classe e selecione Run As -> Spring Boot App. A aplicação será iniciada e estará acessível via HTTP na porta padrão 8080 (e.g., http://localhost:8080).

**6. Detalhamento Técnico das Classes Principais**

Uma explanação aprofundada dos componentes essenciais do projeto esclarece suas responsabilidades e interações:

* **UsuarioFinanceiro.java (Entidade / Modelo de Dados):** Esta classe é o cerne da representação de dados no sistema. Mapeada para a tabela usuario\_financeiro no banco de dados, ela define a estrutura das informações financeiras de cada indivíduo. Inclui atributos como id (identificador único, configurado para autoincremento), nome, rendaMensal, despesasFixas e totalDividas. A implementação de construtores, métodos *getters* e *setters* facilita o acesso e a manipulação dos dados encapsulados.
* **UsuarioFinanceiroRepository.java (Camada de Persistência - Repository):** Como uma interface que estende JpaRepository<UsuarioFinanceiro, Long>, esta classe opera como a ponte direta entre a aplicação e o banco de dados para a entidade UsuarioFinanceiro. Sua herança de JpaRepository elimina a necessidade de codificação manual para operações CRUD básicas, provendo métodos como save, findById, findAll e deleteById de forma abstrata.
* **UsuarioFinanceiroService.java (Camada de Lógica de Negócio - Service):** Esta classe abriga a inteligência central do negócio. Ela coordena as operações de persistência e recuperação de dados através da injeção de dependência de UsuarioFinanceiroRepository. Além das funcionalidades de gerenciamento (listarTodos, buscarPorId, salvar, atualizar, deletar), o método calcularNivelEndividamento é crucial. Este método calcula o percentual de endividamento e, com base em faixas predefinidas, atribui uma classificação de risco e sugere ações financeiras, validando a integridade dos dados de entrada antes do cálculo.
* **UsuarioFinanceiroController.java (Camada de Apresentação - Controller):** Este controlador é o ponto de entrada da API REST. Anotado com @RestController e @RequestMapping("/api/usuarios"), ele define a URL base para todas as operações relacionadas a usuários financeiros. A anotação @CrossOrigin(origins = "\*") é empregada para permitir requisições de diferentes domínios, facilitando a integração com *frontends*. O controlador expõe *endpoints* para todas as operações CRUD (GET para listagem e busca por ID, POST para criação, PUT para atualização e DELETE para exclusão), delegando a execução das lógicas ao UsuarioFinanceiroService. A classe interna UsuarioComRecomendacao é utilizada para padronizar a resposta do *endpoint* de busca por ID, incluindo os dados do usuário e sua recomendação de endividamento.
* **PlataformaEndividamentoApplication.java (Ponto de Entrada da Aplicação):** Esta classe, com a anotação @SpringBootApplication, serve como o ponto de partida para a execução da aplicação Spring Boot. O método main é responsável por iniciar o contexto da aplicação.
* **application.properties (Arquivo de Configuração):** Este arquivo armazena as propriedades essenciais para o funcionamento da aplicação. As configurações de conexão com o banco de dados MySQL (spring.datasource.url, username, password) são cruciais para a comunicação com a base de dados. Além disso, as propriedades spring.jpa.hibernate.ddl-auto=update, spring.jpa.show-sql=true e spring.jpa.properties.hibernate.format\_sql=true configuram o Hibernate para gerenciar o esquema do banco de dados automaticamente e exibir as *queries* SQL no console, respectivamente.

**7. Exemplos de Interação com a API**

Para ilustrar a interação com a API, são apresentados exemplos de requisições HTTP. Ferramentas como Insomnia, Postman ou comandos curl podem ser empregadas para testar os *endpoints*:

* **Criação de um Novo Usuário (Requisição POST):**
  + **URL:** http://localhost:8080/api/usuarios
  + **Método:** POST
  + **Header:** Content-Type: application/json
  + **Corpo da Requisição (JSON):**

JSON

{

"nome": "João Silva",

"rendaMensal": 5000.00,

"despesasFixas": 2000.00,

"totalDividas": 1000.00

}

* **Listagem de Todos os Usuários (Requisição GET):**
  + **URL:** http://localhost:8080/api/usuarios
  + **Método:** GET
* **Busca de Usuário por ID e Análise de Endividamento (Requisição GET):**
  + **URL:** http://localhost:8080/api/usuarios/{id} (onde {id} deve ser substituído pelo ID do usuário desejado)
  + **Método:** GET
* **Atualização de um Usuário Existente (Requisição PUT):**
  + **URL:** http://localhost:8080/api/usuarios/{id} (onde {id} deve ser substituído pelo ID do usuário a ser atualizado)
  + **Método:** PUT
  + **Header:** Content-Type: application/json
  + **Corpo da Requisição (JSON):**

JSON

{

"nome": "João Silva Atualizado",

"rendaMensal": 6000.00,

"despesasFixas": 2500.00,

"totalDividas": 3000.00

}

* **Exclusão de um Usuário (Requisição DELETE):**
  + **URL:** http://localhost:8080/api/usuarios/{id} (onde {id} deve ser substituído pelo ID do usuário a ser removido)
  + **Método:** DELETE

**8. Considerações Finais: Desafios e Evoluções Futuras**

Durante o desenvolvimento deste projeto, diversos desafios foram superados, proporcionando valiosos aprendizados. A configuração e integração do ambiente Spring Boot com Maven e MySQL, bem como o mapeamento de entidades JPA e a utilização do Spring Data JPA para abstrair o acesso ao banco de dados, representaram etapas fundamentais no processo. A estruturação da lógica de negócio na camada de *Service* e a subsequente exposição das funcionalidades através de um *Controller* RESTful, além da customização de respostas da API, foram pontos chave que enriqueceram a experiência de desenvolvimento.

Este projeto consolidou conhecimentos sobre a arquitetura em camadas de aplicações Spring Boot, o uso eficiente de ORMs como o Spring Data JPA para operações CRUD, e a construção de APIs RESTful. Como evoluções futuras, sugere-se a implementação de validações mais robustas nos dados de entrada (utilizando, por exemplo, @Valid e @NotNull), o aprimoramento do tratamento global de exceções para fornecer mensagens de erro mais amigáveis, e o desenvolvimento de um *frontend* interativo que consuma a API. Além disso, a inclusão de mecanismos de autenticação e autorização para proteger os *endpoints* e a criação de testes de unidade e integração mais abrangentes seriam melhorias significativas para a robustez do sistema.

**9. Nota ao Professor Diego Paula**

Prezado Professor Diego Paula,

A aluna Fernanda Pereira da Silva, responsável pela etapa de documentação e testes deste projeto, gostaria de apresentar suas sinceras desculpas por quaisquer imperfeições ou falhas que possam ser identificadas no presente relatório.

É importante esclarecer que a elaboração desta documentação está sendo realizada em um contexto de extrema urgência e durante seu período de trabalho de estágio, o que só foi possível mediante a expressa permissão de sua supervisora. Esta situação excepcional decorreu da pane completa de seu computador pessoal na noite do dia 16 de junho, por volta das 19:30. Todo o material previamente preparado para a documentação e os dados de teste estavam armazenados neste equipamento, que cessou de funcionar, tornando o acesso a tais informações inviável.

O imprevisto não apenas impactou a qualidade e o tempo dedicado à documentação deste projeto, forçando sua elaboração de forma acelerada, mas também impediu a entrega de trabalhos pertinentes à disciplina do Professor João Paulo, que haviam sido solicitados para uma repescagem com prazo final até às 23:59 do mesmo dia 16 de junho.

A aluna reitera seu compromisso com a qualidade acadêmica e expressa seu lamento por qualquer inconveniente causado por esta circunstância fortuita.

Atenciosamente,

Fernanda Pereira da Silva RA: 172317393