Sprint 0 – CSI-29 – Infinity – Money Manager Web

1. Relação dos alunos na equipe

- Gabriel Telles Missailidis
- João Lucas Rocha Rolim
- Othon Daiki Ishiyi
- Samir Nunes da Silva

2. Projeto a ser desenvolvido

2.1. Título

O projeto é intitulado “Infinity - Money Manager Web”.

2.2. Descrição

O sistema será um aplicativo Web e funcionará como um organizador e de finanças pessoais, sendo uma nova visão do projeto de CSI-28. Em CSI-28, foi feita uma aplicação Android na linguagem Kotlin para gestão de finanças, chamada de “Infinity - Money Manager”. Já em CSI-29, a proposta será adaptar a solução para a Web, utilizando no backend o framework Spring para criar APIs RESTful, também na linguagem Kotlin. Por sua vez, o frontend será feito em Javascript, através do framework React.

2.3. Origem

O projeto surgiu da necessidade de um colega de organizar suas finanças pessoais após se tornar aspirante da Força Aérea. Nesse contexto, a equipe concordou em ajudá-lo, oferecendo-o uma solução via software: o Infinity – Money Manager, para organizar e gerir seu dinheiro de maneira mais fácil e intuitiva. Esse colega tornou-se, portanto, o cliente principal da aplicação desenvolvida e foi escutado para a coleta dos requisitos do sistema.

2.4. Público-alvo

O perfil de usuário que se interessaria por uma aplicação de gestão de finanças pessoais, como o Infinity - Money Manager, é notavelmente diversificado. Essa variedade de interesses e necessidades reflete a versatilidade e utilidade do aplicativo. Entre os potenciais usuários, estão aqueles que buscam manter a sanidade financeira em um ambiente domiciliar, controlando gastos, economizando para emergências e planejando a aposentadoria.

Além disso, também inclui iniciantes que desejam adquirir mais conhecimento sobre a gestão de dinheiro, aprender sobre investimentos, entender como otimizar suas finanças e aproveitar ao máximo seu dinheiro.

Outro grupo de usuários em potencial são aqueles que têm objetivos monetários específicos, como economizar para uma viagem, comprar um carro, adquirir uma casa, pagar dívidas ou investir na educação dos filhos. Eles reconhecem a importância de um planejamento financeiro sólido para alcançar esses objetivos e estão em busca de uma ferramenta que os ajude a traçar um caminho claro para atingir essas metas.

- **Jovens Profissionais**

Recém-formados e jovens profissionais muitas vezes procuram aplicativos de gerenciamento financeiro para obter controle sobre suas finanças, acompanhar despesas e começar a economizar ou investir. Eles podem usar o aplicativo para criar orçamentos, monitorar suas receitas e gastos e planejar marcos financeiros futuros, como comprar uma casa ou pagar empréstimos estudantis.

- **Famílias**

Os responsáveis familiares podem usar aplicativos de gerenciamento financeiro para manter o orçamento doméstico, gerenciar despesas relacionadas a possíveis filhos, economizar para a educação e se preparar para despesas inesperadas. Esses usuários também podem usar recursos como lembretes de contas e compartilhamento de despesas para manter as finanças familiares organizadas.

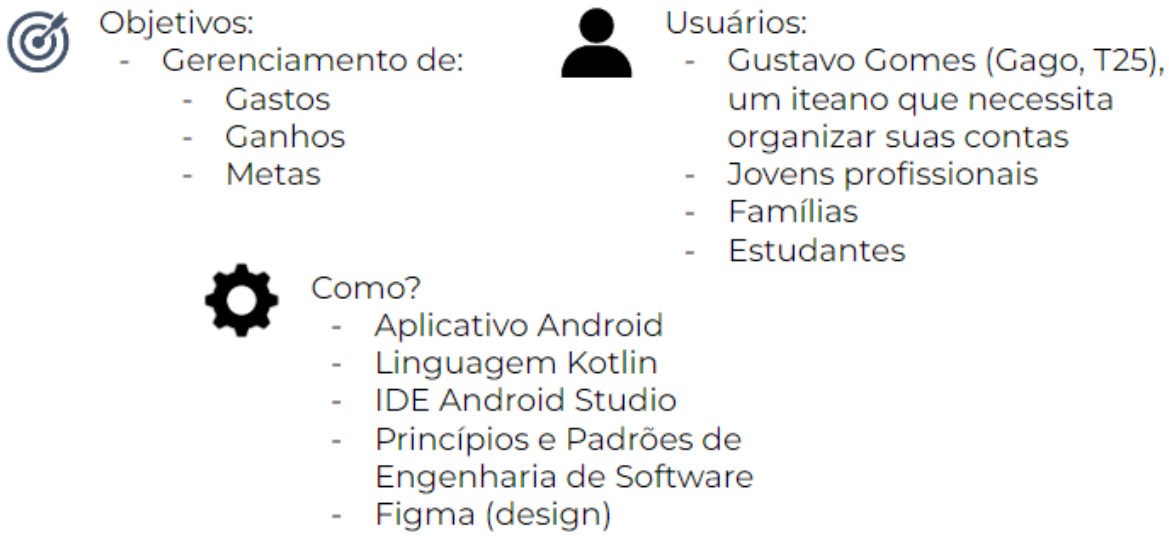
- **Estudantes**

Estudantes universitários e jovens adultos podem se beneficiar de aplicativos de gerenciamento financeiro para aprender sobre responsabilidade financeira, orçamentar mensalidades e despesas de subsistência e evitar o acúmulo de dívidas. Esses usuários podem seconcentrar em monitorar seus gastos discricionários.

2.5. O que já foi implementado no projeto de CSI-28

Em CSI-28, foi entregue um protótipo de aplicativo Android para gerenciamento de finanças. Nele, foram implementados vários passos, os quais são mostrados a seguir:

- Objetivos, usuários e como:



- Documento de requisitos:

Capítulo

Requisitos funcionais (casos de uso) 2

Gestão de Finanças

Esta subseção descreve os requisitos relacionados às ferramentas de gestão de finanças do sistema. Convém agrupá-los em uma mesma subseção pois estão atrelados a uma mesma tela.

[RF001] Tela de Gestão de Finanças

Ao inicializar o aplicativo, o usuário será exposto à tela principal de Gestão de Finanças, na qual será possível visualizar a situação financeira do mês atual e dos meses anteriores, adicionar gastos e ganhos e acompanhar metas financeiras. Haverá um botão de acesso rápido no canto inferior esquerdo da tela para acessar funcionalidades mais recorrentes.

Ator: Usuário.

Entradas e pré-condições: O usuário inicializa o aplicativo *Infinity - Money Manager*.

Saídas e pós-condições: A tela de Gestão de Finanças é apresentada.

Prioridade: ❒ Essencial ☐ Importante ☐ Desejável

Capítulo

Requisitos não funcionais 3

Usabilidade

Esta seção descreve os requisitos não funcionais associados à facilidade de uso da interface com o usuário, material de treinamento e documentação do sistema.

[NF001] Botões de fácil entendimento

Os botões do aplicativo, sejam para fazer uma determinada função, sejam para destinar o usuário a outra tela, devem ter suas atribuições facilmente compreensíveis pelo usuário, tornando a interface do aplicativo de simples utilização.

Prioridade: ❒ Essencial ☐ Importante ☐ Desejável

[NF002] Uso natural de cores e símbolos

O uso de cores e símbolos no aplicativo deve facilitar sua usabilidade, no sentido de proporcionar ao usuário uma tela limpa, de fácil compreensão e agradável à visão. Nesse contexto, cores e símbolos diferentes ou exagerados não devem ser usados em excesso, evitando assim a poluição visual das interfaces gráficas.

Prioridade: ❒ Essencial ☐ Importante ☐ Desejável

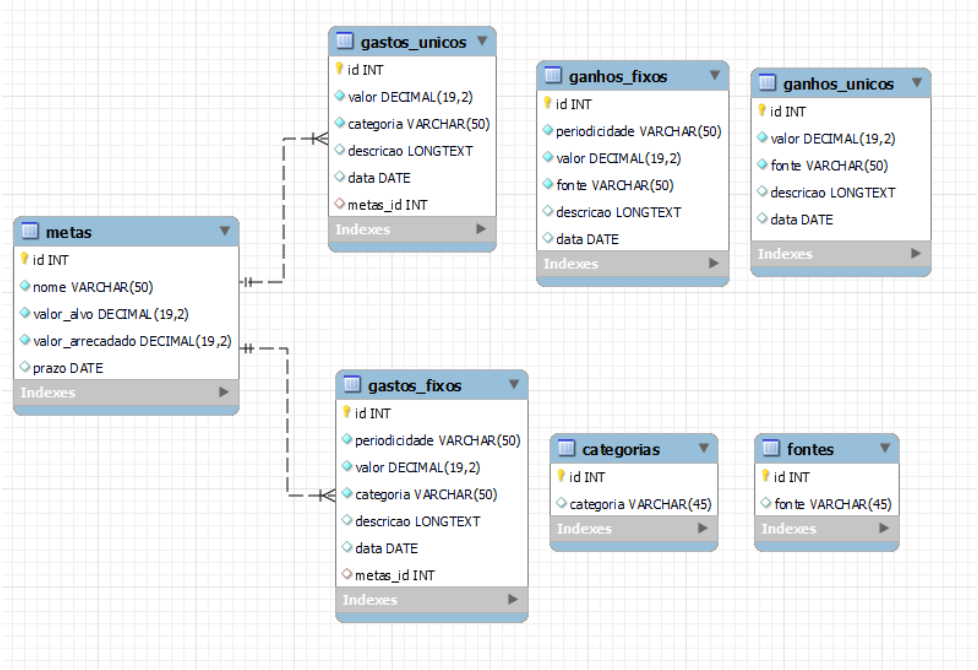
Confiabilidade

Esta seção descreve os requisitos não funcionais associados à freqüência, severidade de falhas do sistema e habilidade de recuperação das mesmas, bem como à corretude do sistema.

[NF003] Robustez a falhas de memória

Como o sistema possui salvamento apenas local, seus dados, se perdidos, não poderão ser recuperados. Nesse sentido, deve evitar que perdas de informações dadas pelos usuários ao

- Banco de dados:



- Classes do banco de dados:

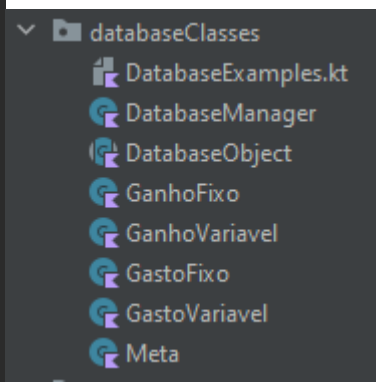
```kotlin
package com.example.infinitymoneymanager.databaseClasses

import java.sql.PreparedStatement

Samir Nunes da Silva
abstract class DatabaseObject{
    protected abstract val name: String
    protected abstract val sqlTable: String
    protected abstract val sqlColumns: String

    Samir Nunes da Silva
    abstract fun setQueryVariables(query: PreparedStatement)

    Samir Nunes da Silva
    fun getObjectName(): String {return name}
    Samir Nunes da Silva
    fun getSqlTableName(): String {return sqlTable}
    Samir Nunes da Silva
    fun getSqlColumnsNames(): String {return sqlColumns}
}
```



- Operações do banco de dados:

Banco de Dados

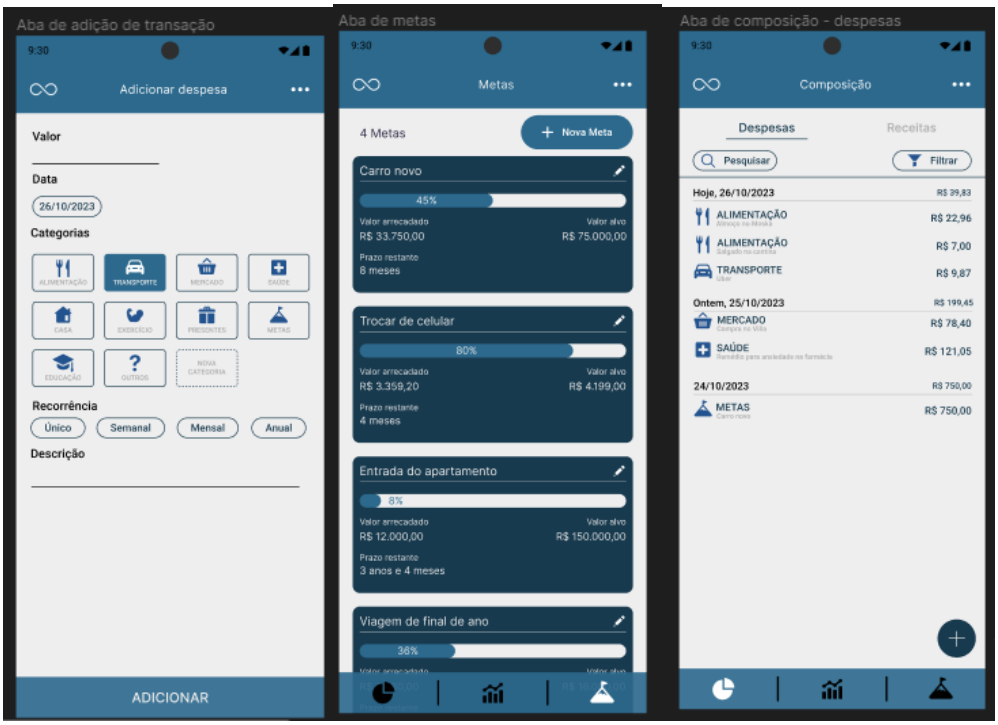
Operações:
- Insert
- Delete
- Select

```
package com.example.infinitymoneymanager.databaseClasses

import java.sql.Connection
import java.sql.DriverManager
import java.sql.PreparedStatement
import java.sql.ResultSet

Samir Nunes da Silva
class DatabaseManager{
    Samir Nunes da Silva
    companion object {
        private var connection: Connection? = null
        Samir Nunes da Silva
        @JvmStatic
        fun openConnection() {
            connection = DriverManager.getConnection(
                url: "jdbc:mysql://localhost/infinity",
                user: "root",
                password: "infinity"
            )
            println("Connection with database opened successfully.")
        }
```

- Artes no Figma:

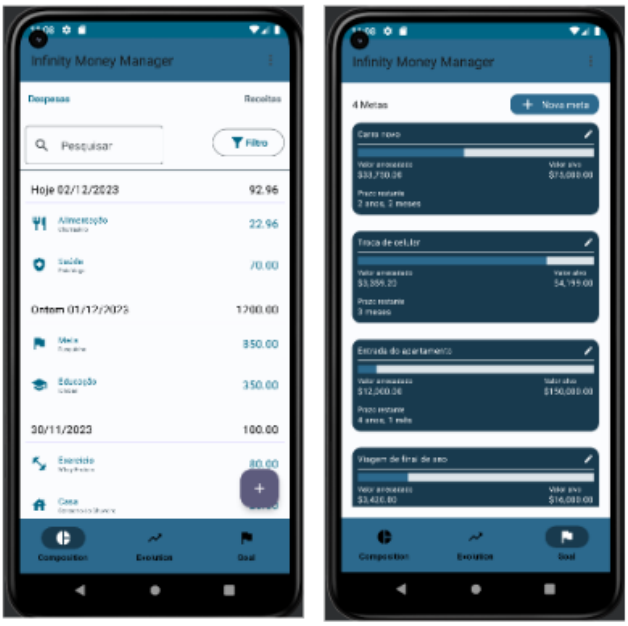


- Frontend para Android:

Adotou-se o framework Jetpack Compose para o desenvolvimento da UI.

Frontend

- Responsável pela Interface com o Usuário (UI)



- Backend para Android:

```kotlin
@RequestMapping("/finance")
class FinanceController(private val financeService: FinanceService) {

    @PostMapping("/gastos")
    fun createGasto(@RequestBody gastoDto: GastoDto): GastoFixo {
        println(gastoDto.categoria)
        return financeService.createGasto(gastoDto)
    }

    @PostMapping("/metas")
    fun createMeta(@RequestBody metaDto: MetaDto): Meta {
        return financeService.createMeta(metaDto)
    }

    @PostMapping("/gastos-variaveis")
    fun createGastoVariavel(@RequestBody dto: GastoVariavelDto): GastoVariavel {
        return financeService.createGastoVariavel(dto)
    }

    @PostMapping("/ganhos-fixos")
    fun createGanhoFixo(@RequestBody dto: GanhoFixoDto): GanhoFixo {
        return financeService.createGanhoFixo(dto)
    }

    @PostMapping("/ganhos-variaveis")
    fun createGanhoVariavel(@RequestBody dto: GanhoVariavelDto): GanhoVariavel {
        return financeService.createGanhoVariavel(dto)
    }

    @GetMapping("/gastos")
    fun getAllGastos(): ResponseEntity<List<Any>> {
        val gastos = financeService.getAllGastos()
        return ResponseEntity.ok(gastos)
    }

    @GetMapping("/ganhos")
    fun getAllGanhos(): ResponseEntity<List<Any>> {
        val ganhos = financeService.getAllGanhos()
        return ResponseEntity.ok(ganhos)
    }
```

2.6. Visão Geral das Funcionalidades a Serem Implementadas

Em linhas gerais, as funcionalidades do Infinity - Money Manager Web serão as seguintes:

- Adição, remoção ou alteração de valores de ganhos, rendimento e gastos, e classificação desses valores em categorias;
- Definição de metas e visualização daquelas em progresso;
- Visualização gráfica dos dados financeiros;

Quanto ao que o sistema não irá fazer, tem-se:

- Armazenamento de informações financeiras e pessoais do usuário fora do escopo local;
- Integração com sistemas bancários;
- Realização de operações financeiras, como transferência e recebimento de dinheiro;
- Previsão do sucesso ou fracasso de investimentos;

Quanto à interação com outros sistemas, o aplicativo será independente e totalmente autocontido, de forma que apenas seus módulos desenvolvidos serão necessários e suficientes para o uso de todas suas funcionalidades por parte do usuário.

2.7. Backlog – Requisitos Funcionais

Obs: RFXXX refere-se ao requisito funcional de número XXX.

[RF001] Tela de Gestão de Finanças

Ao acessar a aplicação web, o usuário será exposto à tela principal de Gestão de Finanças, na qual será possível visualizar a situação financeira do mês atual e dos meses anteriores, adicionar gastos e ganhos e acompanhar metas financeiras. Haverá um botão de acesso rápido para acessar funcionalidades mais recorrentes.

Ator: Usuário.

Entradas e pré-condições: O usuário inicializa o aplicativo *Infinity - Money Manager.*

Saídas e pós-condições: A tela de Gestão de Finanças é apresentada.

[RF002] Criação de gastos e ganhos únicos

O usuário deverá ser capaz de criar gastos e ganhos únicos. Para cadastrar o fluxo monetário, haverá campos nos quais o usuário informa o valor monetário envolvido, a categoriado gasto/ganho e, opcionalmente, uma descrição.

Ator: Usuário.

Entradas e pré-condições: O usuário está na tela de Gestão de Finanças e, no botão de acesso rápido, seleciona criar um gasto/ganho único.

Saídas e pós-condições: As mudanças são registradas e armazenadas localmente em um banco de dados.

[RF003] Adição de gastos e ganhos fixos

O usuário deverá ser capaz de criar gastos e ganhos fixos (periódicos). Para cadastrar o

fluxo monetário, haverá campos nos quais o usuário informa, caso este seja fixo, o valor monetário envolvido, a categoria do gasto/ganho, a periodicidade, e, opcionalmente, uma descrição.

Ator: Usuário.

Entradas e pré-condições: O usuário está na tela de Gestão de Finanças e clica para adicionar um gasto/ganho recorrente.

Saídas e pós-condições: As mudanças são registradas e armazenadas localmente em um banco de dados.

[RF004] Visualização dos ganhos e gastos

Na tela principal deve haver uma seção de resumo, na qual o usuário é informado das quantidades total arrecadada e gasta no mês, além da porcentagem economizada. O usuário também deverá ser capaz de acessar a íntegra dos fluxos financeiros e filtrar os resultados por tipo de fluxo (gasto ou ganho), periodicidade do fluxo financeiro e categoria. O usuário também deverá poder alternar, na mesma tela, entre diferentes meses de análise.

Ator: Usuário.

Entradas e pré-condições: O usuário está no card reservado para visualização de gastose ganhos.

Saídas e pós-condições: Os gastos e ganhos são exibidos conforme os filtros selecionados.

[RF005] Edição e remoção dos ganhos e gastos

Também na visualização dos gastos e ganhos, o usuário deverá ser capaz de editar ou remover algum fluxo monetário.

Ator: Usuário.

Entradas e pré-condições: O usuário está no card reservado para visualização de gastose seleciona algum fluxo monetário exibido.

Saídas e pós-condições: A edição ou remoção do gasto selecionado será registrado e o banco de dados será atualizado localmente.

[RF006] Criação, edição e remoção de metas financeiras

O usuário, na tela de Gestão de Finanças, deverá ser capaz de criar, editar e remover metas financeiras. O usuário deverá informar o nome da meta, o valor monetário almejado e o prazo para atingir o objetivo.

Ator: Usuário.

Entradas e pré-condições: O usuário está no card reservado para metas, contido na tela de Gestão de Finanças, e clica em adicionar uma nova meta ou seleciona uma meta já estabelecida para editá-la ou removê-la.

Saídas e pós-condições: A criação, edição ou remoção da meta será registrada em um banco de dados local.

[RF007] Registrar dinheiro reservado para metas financeiras

Para cada meta criada, o usuário poderá selecionar, dentre as opções de categorias degasto, a opção de “dinheiro economizado” para a meta.

Ator: Usuário.

Entradas e pré-condições: O usuário, ao adicionar um gasto, seleciona a categoria dereservar dinheiro para uma das metas.

Saídas e pós-condições: O valor cadastrado é armazenado em um banco de dados local.

[RF008] Visualizar progresso de metas financeiras

Na tela de Gestão Financeira, o usuário poderá visualizar, para cada meta, um gráfico que compara a quantia economizada com a quantia faltante. Além disso, será exibida a média mensal que o usuário deve reservar para atingir a meta dentro do prazo estabelecido.

Ator: Usuário.

Entradas e pré-condições: O usuário está no card, contido na tela de Gestão de Finanças, reservado para metas.

Saídas e pós-condições: Os gráficos e demais resultados são exibidos conforme dados cadastrados para a meta e quantia já economizada para tal fim.

3. Sprint 01

Na Sprint 01 serão prototipados os requisitos RF002, RF003, RF005 e RF006 do backlog do produto por meio dos seguintes passos:

- Setup do banco de dados local no MySQL Workbench (já feita no projeto anterior);
- Setup do ambiente de programação;
- Criação da arte das telas no Figma;
- Criação do protótipo da tela de Adicionar Despesas;
- Criação do protótipo da tela de Metas;
- Criação do protótipo da tela de Filtros;

4. Painel Kanban

Foi criado um quadro Kanban para o projeto no software Jira, o qual é mostrado a seguir. Na coluna TO DO, foram dispostas algumas tarefas iniciais. Elas se referem a tarefas que serão feitas após a Sprint 01.

Já na coluna IN PROGRESS, foram colocadas as tarefas referentes à Sprint 01. São elas:

- Setup do ambiente de programação;
- Criação das telas no Figma;
- Fazer o protótipo do serviço AdicionarDespesasScreen;
- Fazer o protótipo do serviço MetasScreen;
- Fazer o protótipo do serviço FiltrosScreen;

Por fim, na coluna DONE foi colocada a tarefa de Setup do banco de dados MySQL, que já havia sido feita.