GRUPO DE ESTUDO DE EDUCAÇÃO À DISTÂNCIA / CETEC
TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS – MODALIDADE EaD

CLAUDIA DE JESUS DANTAS

CRISTINE AGUIAR DE MACEDO

GABRIEL TAYLOR DE SOUSA SILVA

ROBSON LUIZ SOARES DA CONCEIÇÃO

WAGNER SOUZA GOMES SILVA

APLICATIVO DE CONTROLE DE GASTOS NO KODULAR: Save Money

São Paulo

2023

CLAUDIA DE JESUS DANTAS
CRISTINE AGUIAR DE MACEDO
GABRIEL TAYLOR DE SOUSA SILVA
ROBSON LUIZ SOARES DA CONCEIÇÃO
WAGNER SOUZA GOMES SILVA

APLICATIVO DE CONTROLE DE GASTOS NO KODULAR: Save Money

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas – modalidade EaD, orientado pelo Prof. José Mendes da Silva Neto, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas.

São Paulo
2023

RESUMO

Este trabalho apresenta o desenvolvimento do aplicativo "Save Money" utilizando a plataforma Kodular, com o objetivo de proporcionar aos usuários uma ferramenta simples e eficaz para o controle de gastos e receitas. O aplicativo visa atender à necessidade crescente de soluções acessíveis que auxiliem na organização financeira pessoal. A interface intuitiva do aplicativo permite que os usuários registrem suas transações financeiras de forma descomplicada, promovendo uma conscientização sobre padrões de consumo e contribuindo para uma gestão mais eficiente dos recursos financeiros. O "Save Money" apresenta de forma clara e organizada o total de receitas, gastos e saldo líquido, oferecendo aos usuários uma visão abrangente de sua situação financeira. Ao longo do desenvolvimento, foram aplicados conhecimentos teóricos adquiridos ao longo do curso, consolidando habilidades de programação, design de interface e resolução de problemas. Considerando as conclusões, destaca-se a importância do aplicativo como uma contribuição tangível para o campo do desenvolvimento de sistemas e aplicativos móveis. Recomendações para futuras versões incluem a implementação de recursos avançados, aprimoramento da segurança, coleta de feedback dos usuários e a possível expansão para outras plataformas. O "Save Money" representa uma solução prática para o desafio comum do controle financeiro pessoal, destacando a relevância contínua do aprimoramento tecnológico na sociedade atual.

Palavras-chave: Controle Financeiro Pessoal. Kodular. Aplicativo móvel. Conscientização Financeira. Desenvolvimento de sistemas.

ABSTRACT

This work presents the development of the "Save Money" app using the Kodular platform, aiming to provide users with a simple and effective tool for expense and income management. The app addresses the growing need for accessible solutions that assist in personal financial organization. The app's intuitive interface allows users to record their financial transactions effortlessly, promoting awareness of consumption patterns and contributing to more efficient resource management. "Save Money" presents the total income, expenses, and net balance in a clear and organized manner, offering users a comprehensive view of their financial situation. Throughout development, theoretical knowledge acquired during the course was applied, consolidating skills in programming, interface design, and problem-solving. Considering the conclusions, the significance of the app as a tangible contribution to the field of system and mobile app development is highlighted. Recommendations for future versions include implementing advanced features, enhancing security, gathering user feedback, and potentially expanding to other platforms. "Save Money" represents a practical solution to the common challenge of personal financial control, underscoring the ongoing relevance of technological advancement in contemporary society.

Keywords: Personal Financial Management. Kodular. Mobile App. Financial Awareness. System Development.

SUMÁRIO

1. INTRODUÇÃO

No cenário atual, a gestão eficiente das finanças pessoais é essencial para a estabilidade financeira e o bem-estar dos indivíduos. Nesse contexto, surge a necessidade de desenvolver um aplicativo que possa auxiliar os usuários no registro de suas despesas e receitas.

O principal objetivo deste projeto é criar um aplicativo que permita aos usuários registrarem suas despesas e receitas de forma simples e eficaz. O aplicativo não visa gerar relatórios complexos ou análises detalhadas, mas sim oferecer uma ferramenta prática para o registro de informações financeiras.

A justificativa para o desenvolvimento deste aplicativo está fundamentada na importância da gestão financeira pessoal. Um bom controle financeiro pode contribuir significativamente para a saúde financeira dos indivíduos, auxiliando-os a tomar decisões informadas sobre seus gastos e economias. A simplicidade e acessibilidade deste aplicativo o tornam uma ferramenta valiosa para pessoas de todas as idades e níveis de conhecimento financeiro.

O aplicativo permitirá aos usuários registrar suas despesas e receitas de maneira organizada. Não incluirá funcionalidades avançadas, como geração de relatórios complexos ou análises financeiras detalhadas. Em vez disso, sua principal função será fornecer uma interface intuitiva e fácil de usar para o registro de informações financeiras pessoais.

A equipe de desenvolvimento deste projeto segue a metodologia ágil Scrum. Ela é composta por seis membros, cada um desempenhando um papel específico. Claudia de Jesus Dantas atua como Product Owner (P.O.), Wagner Souza Gomes Silva assume o papel de Scrum Master, e os membros da equipe de desenvolvimento incluem Cristine Aguiar de Macedo, Gabriel Taylor de Souza Silva e Robson Luiz Soares da Conceição.

As partes interessadas neste projeto abrangem os seguintes grupos:

Usuários Finais: Pessoas que utilizarão o aplicativo para registrar suas despesas e receitas.

Integrantes do Projeto: Alunos envolvidos no desenvolvimento do aplicativo.

Professores: Professores do curso que avaliarão o progresso e a qualidade do projeto.

A premissa fundamental deste projeto é contribuir para a melhoria da saúde financeira das pessoas, promovendo o controle e a gestão eficaz de suas finanças pessoais por meio da disponibilização de uma alternativa eletrônica para o registro de informações financeiras.

O desenvolvimento do aplicativo será realizado na plataforma Kodular, o que limita sua compatibilidade a dispositivos Android. Além disso, o aplicativo utilizará um banco de dados SQLite para o armazenamento de dados. A escolha da plataforma Kodular e do banco de dados SQLite é respaldada pela ideia de que soluções simples podem ser altamente eficazes. Como Larry Wall, criador da linguagem de programação Perl, afirmou, "a simplicidade é a virtude de design mais importante".

O aplicativo permitirá o registro de despesas e receitas de forma eficiente. Não são previstas funcionalidades avançadas ou relatórios complexos, portanto, não são esperadas mudanças significativas no escopo. Em caso de problemas ou falhas, a equipe de desenvolvimento tomará as medidas necessárias para corrigir essas questões.

Autores como Tom DeMarco e Timothy Lister, autores de "Peopleware", destacam a importância do gerenciamento de riscos em projetos de software. Portanto, durante o desenvolvimento do aplicativo, serão adotadas práticas ágeis de gerenciamento de projetos, com ênfase na metodologia Scrum. Isso permitirá a rápida identificação e mitigação de riscos à medida que surgirem.

O aplicativo, mesmo sendo simples em sua proposta, desempenhará um papel importante na vida dos usuários, facilitando o controle de suas finanças pessoais de forma acessível e eficaz.

2. METODOLOGIA

Iremos trabalhar com o software Kodular, que foi desenvolvido por pesquisadores do MIT (Instituto de Tecnologia de Massachusetts) como parte do projeto MIT App Inventor, é uma plataforma de código aberto que visa facilitar a criação de aplicativos móveis através da programação em blocos, onde é possível criar aplicativos para celular de maneira simples a mais complexos sem o conhecimento avançado na programação em Java, Swift, Kotlin entre outras, só para exemplificar.

Utilizando Kodular também trabalharemos com o público voltado para ensino, como é uma linguagem muito voltada para iniciantes na área de tecnologia, podemos divulgar o aplicativo pelo seu intuito de fomentar o conhecimento em gestão financeira e em desenvolvimento mobile, por isso, se faz interessante para professores e alunos a aquisição do Save Money, que será disponibilizado de maneira gratuita e fácil acesso ao ser utilizado por android.

2.1. Requisitos funcionais e não funcionais

O RF1 estabelece que o usuário irá possuir um login para acessar o sistema, com o objetivo de garantir maior segurança e privacidade no uso do aplicativo que gerenciará seus dados financeiros. Essa medida de segurança é fundamental para proteger as informações pessoais e financeiras dos usuários contra acesso não autorizado.

Ao implementar um sistema de login, o aplicativo vai exigir credenciais de nome de usuário e senha, esses métodos de autenticação irão aumentam mais a segurança.

Além disso, ao garantir que apenas usuários autorizados tenham acesso aos

dados do aplicativo, os dados financeiros dos usuários, serão protegidos contra possíveis ameaças de segurança. Isso ajuda a construir confiança entre o usuário e o aplicativo, essencial para o sucesso de qualquer serviço de gerenciamento financeiro.

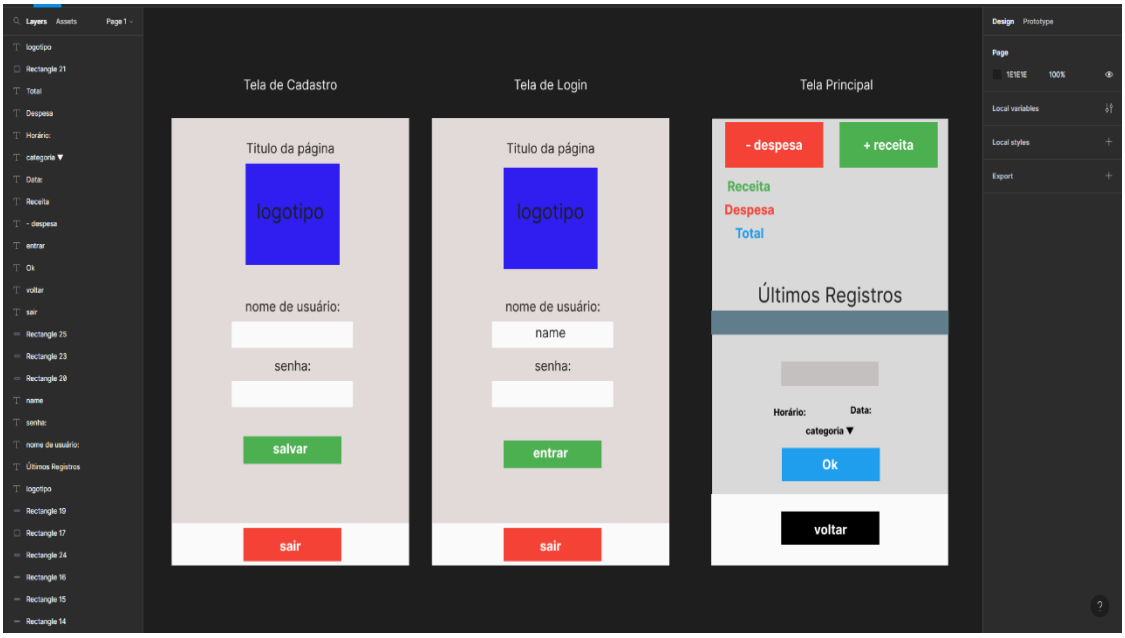
Já o RNF1 do aplicativo especifica que não deverá ser disponibilizada a opção de exportar dados do banco de dados para outros tipos de arquivos, como Excel, PDF ou TXT. Essa restrição visa manter a integridade e a segurança dos dados armazenados no aplicativo, impedindo que informações sensíveis sejam facilmente extraídas e compartilhadas em formatos que possam comprometer a confidencialidade ou ser usadas de maneira inadequada. Essa decisão contribui para garantir a privacidade e a proteção dos dados dos usuários, alinhando-se com as práticas recomendadas de segurança da informação.

3. DESENVOLVIMENTO

3.1. Mockup de Telas - Figma

A parte de desenvolvimento de Front-end começou com o desenho inicial das telas na ferramenta Figma, o link está anexo nesse documento.

Imagem 1 – Desenho inicial das telas

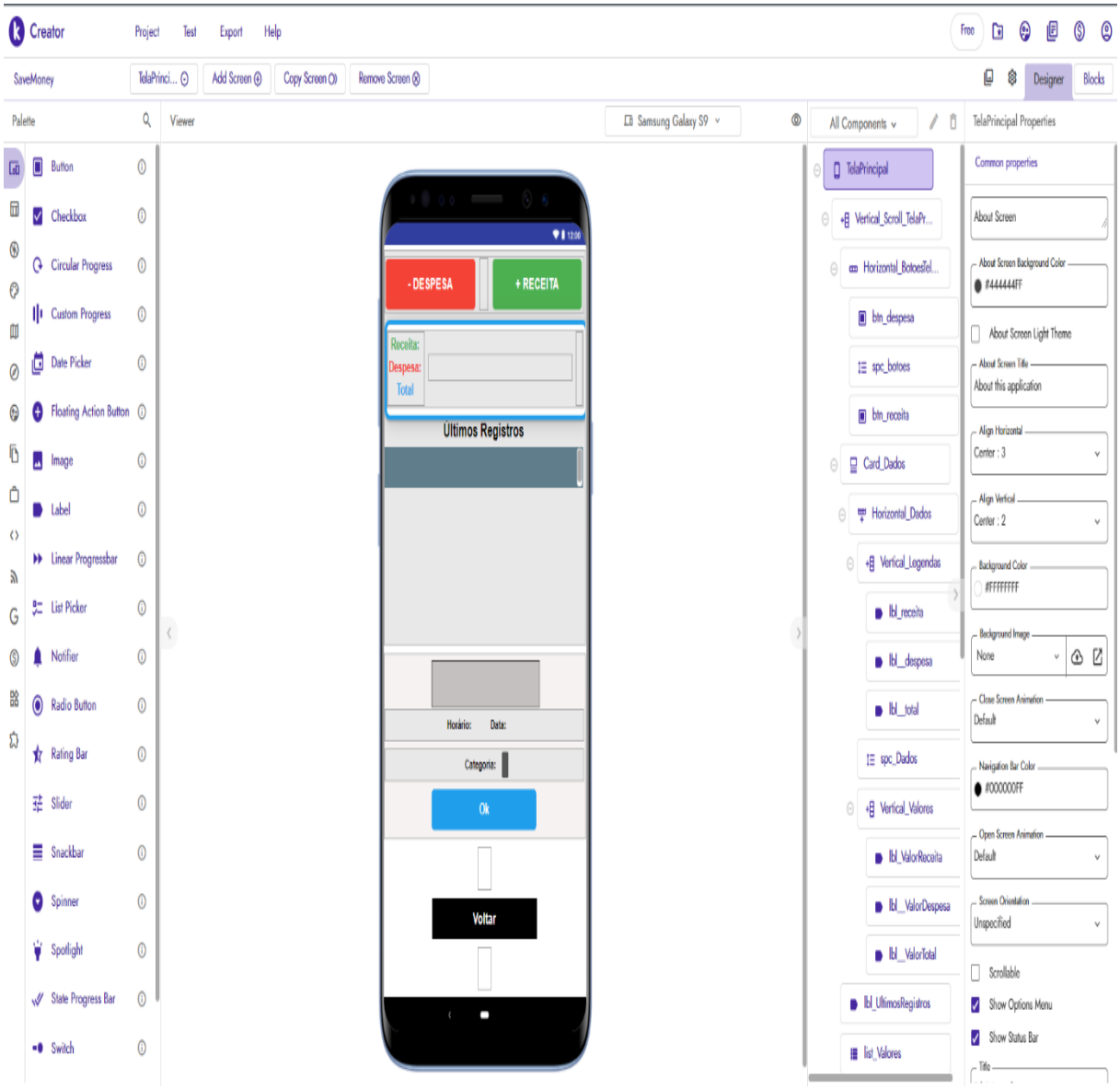


Fonte: Dos próprios autores, 2023

Estabelecido as cores primárias que seriam trabalhadas para deixar o aplicativo mais intuitivo, o verde remetendo a acesso ou no caso do botão receita, algo positivo. O vermelho remete mais a local de saída, aviso e por isso foi escolhido para os botões sair da tela inicial e o preto destacando o voltar para a tela de login. No caso do botão despesa, “estar no vermelho” é um dito popular que remete ao negativo em relação as finanças. O azul do total e do botão “Ok” que insere o valor é uma cor que indica inteligência, então trabalhar com valores que serão somados ou subtraídos aqui, inseridos pelo usuário é para remeter sabedoria, e ao resto do designer a opção de deixar cinza em tonalidades diferentes foi estabelecida para o excesso de cores não desviar o foco do aplicativo e trazer sobriedade no visual, por isso utilizando a psicologia das cores estabelecemos cada uma dessas cores no projeto.

3.1. Mockup de Telas – Versão 2.1.

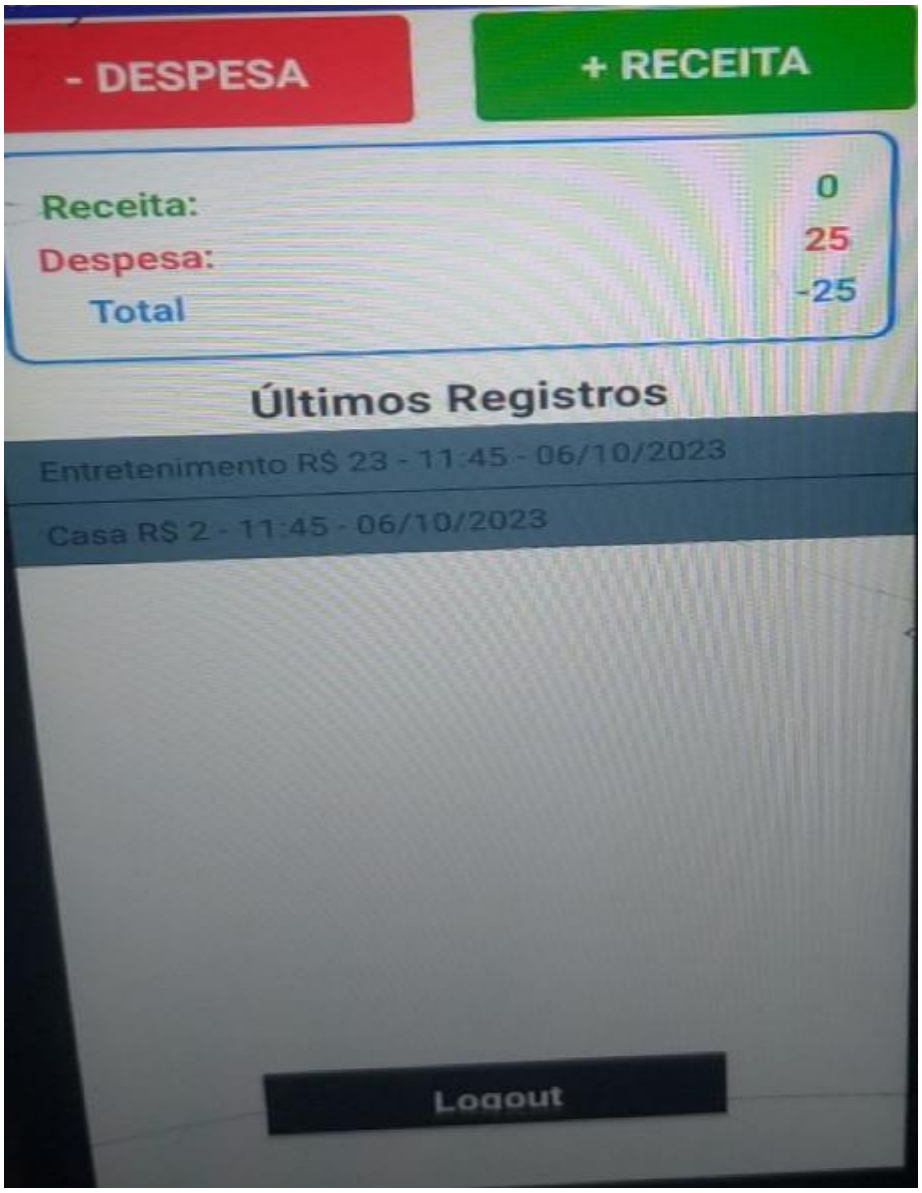
Imagem 2 – Mockup da tela principal versão 2.1



Fonte: Dos próprios autores, 2023

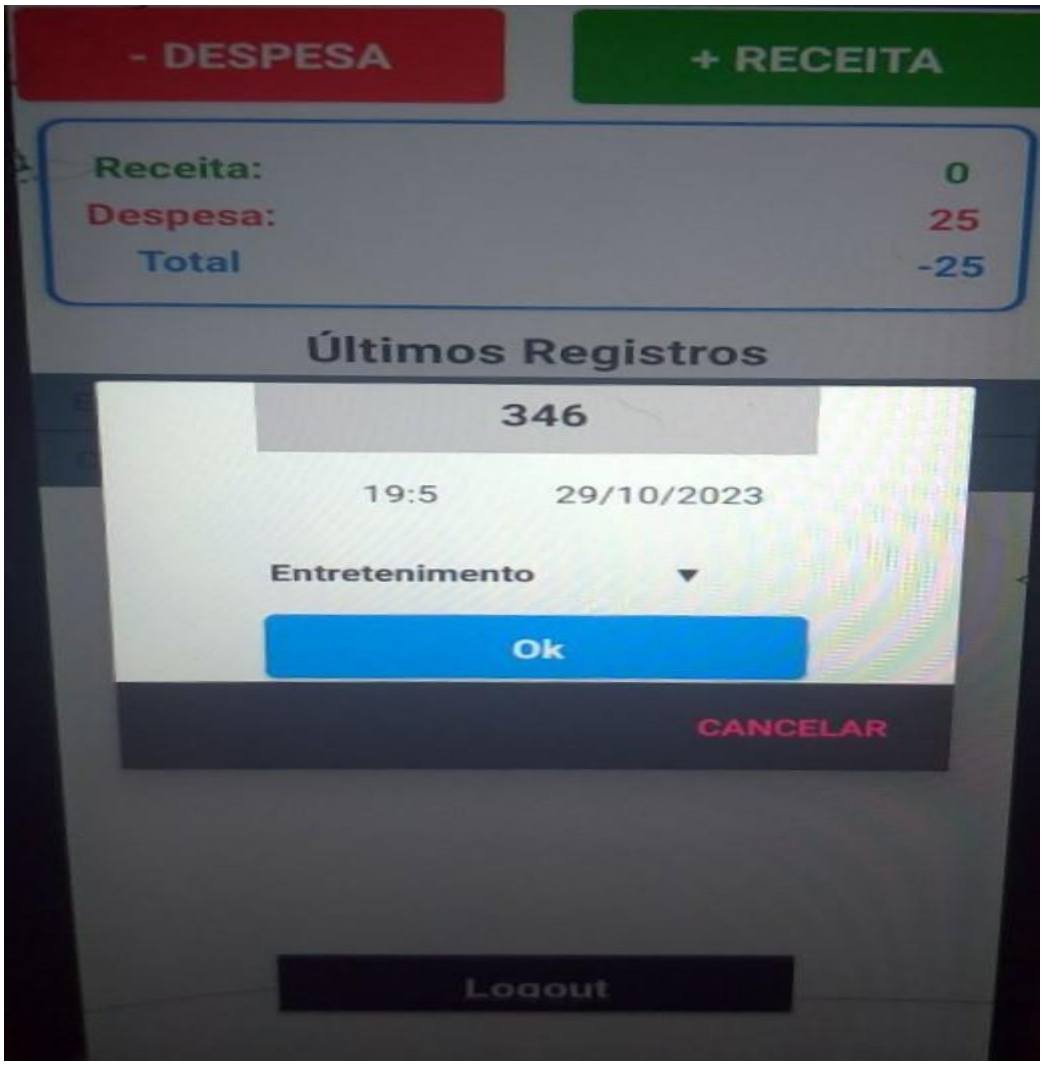
3.2. Mockup de Telas – No dispositivo móvel – Versão 2.2.

Imagem 3 – Mockup da tela principal versão 2.2



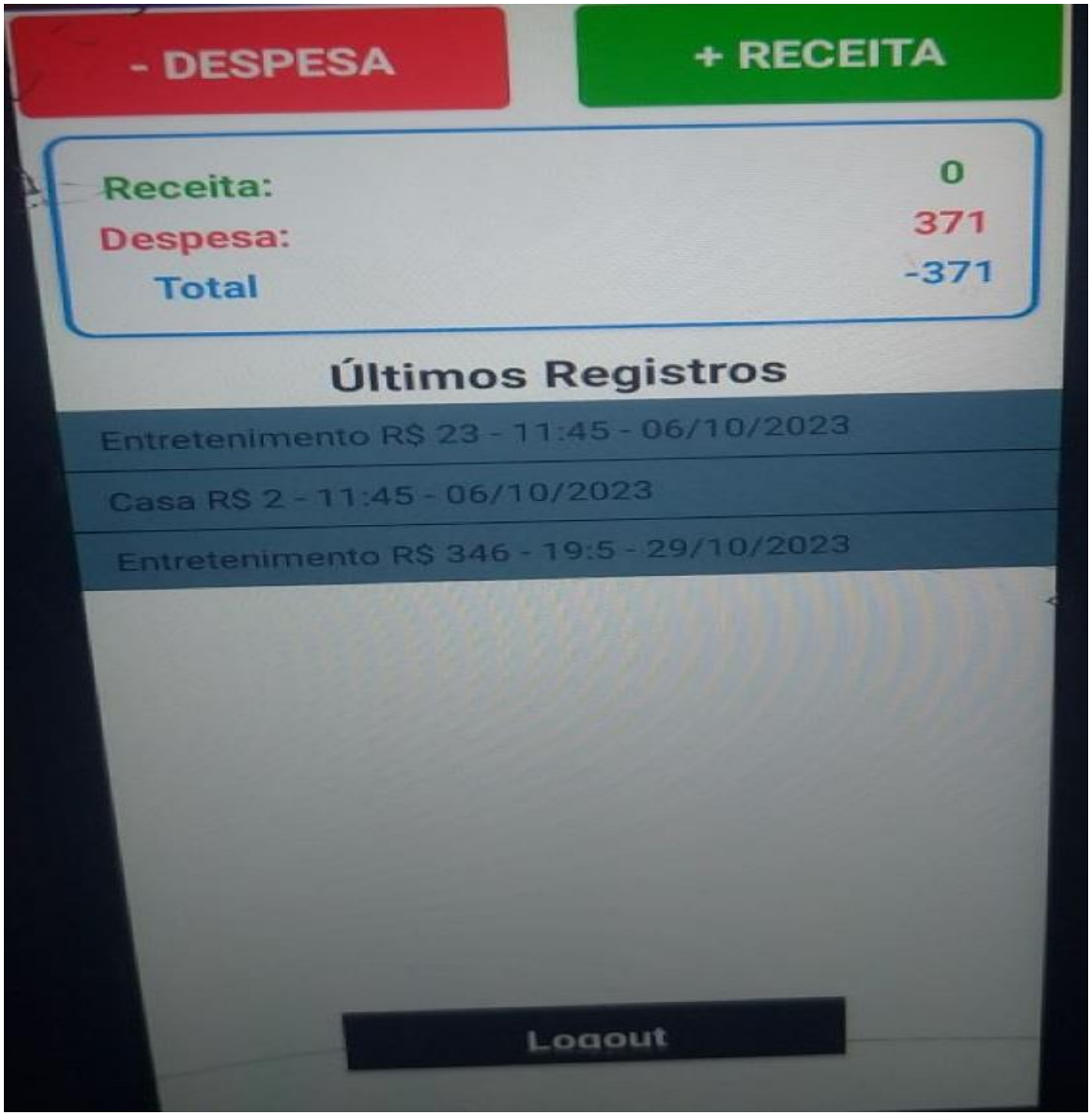
Fonte: Dos próprios autores, 2023

Imagem 4 – Mockup da tela principal – registrando - versão 2.2



Fonte: Dos próprios autores, 2023

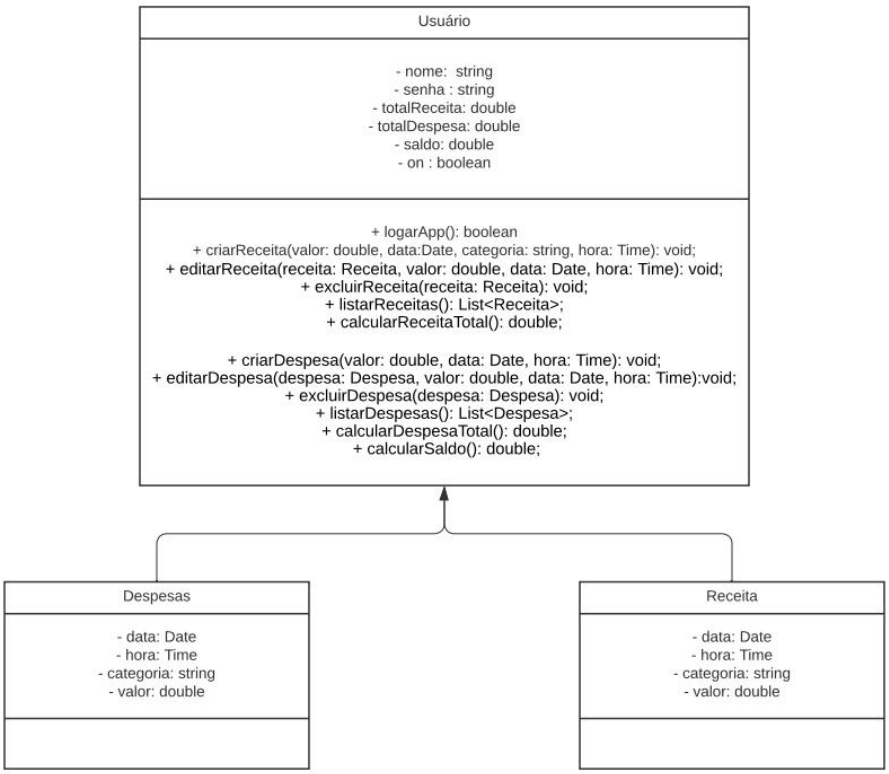
Imagem 4 – Mockup da tela principal – exibição do registro - versão 2.2



Fonte: Dos próprios autores, 2023

3. 2. Classe UML

Imagem 5 – Diagrama de Classe UML



Fonte: Dos próprios autores, 2023

A classe “Usuario” é a classe principal que representa quem de fato irá utilizar nosso aplicativo e como funcionará seu acesso no mesmo. Aqui estão mais detalhes sobre os atributos e métodos:

Atributos:

- nome: String: Armazena o nome do usuário que está usando o aplicativo.
- senha: String: Armazena a senha do usuário.
- totalReceita: double: Mantém o valor total de todas as receitas do usuário.
- totalDespesa: double: Mantém o valor total de todas as despesas do usuário.
- saldo: double: Armazena o saldo atual do usuário.
- on: boolean: É um marcador booleano que indica se o usuário está logado no aplicativo ou não.

Métodos:

- logarApp(): boolean: Esse método é usado para fazer login no aplicativo. Ele retorna true se o login for bem-sucedido e false caso contrário.
- criarReceita(valor: double, data: Date, categoria: String, hora: Time): void: Cria uma nova receita com os detalhes fornecidos, como valor, data, categoria e hora.
- editarReceita(receita: Receita, valor: double, data: Date, hora: Time): void: Permite editar os detalhes de uma receita existente com base no objeto Receita fornecido.
- excluirReceita(receita: Receita): void: Remove uma receita existente com base no objeto Receita fornecido.
- listarReceitas(): List<Receita>: Retorna uma lista de todas as receitas do usuário.
- calcularReceitaTotal(): double: Calcula o valor total de todas as receitas do usuário.
- criarDespesa(valor: double, data: Date, hora: Time): void: Cria uma nova despesa com os detalhes fornecidos, como valor, data e hora.
- editarDespesa(despesa: Despesa, valor: double, data: Date,

hora: Time): void: Permite editar os detalhes de uma despesa existente com base no objeto Despesa fornecido.

- excluirDespesa(despesa: Despesa): void: Remove uma despesa existente com base no objeto Despesa fornecido.
- listarDespesa(): List<Despesa>: Retorna uma lista de todas as despesas do usuário.
- calcularDespesaTotal(): double: Calcula o valor total de todas as despesas do usuário.
- calcularSaldo(): double: Calcula o saldo atual do usuário com base nos valores de receita e despesa.

A classe Despesa representa os detalhes de uma despesa financeira. Ela possui os seguintes atributos:

- data: Date: Armazena a data da despesa.
- hora: Time: Armazena a hora da despesa.
- categoria: String: Indica a categoria à qual a despesa pertence.
- valor: double: Representa o valor da despesa.

A classe Receita representa os detalhes de uma receita financeira. Ela possui os seguintes atributos:

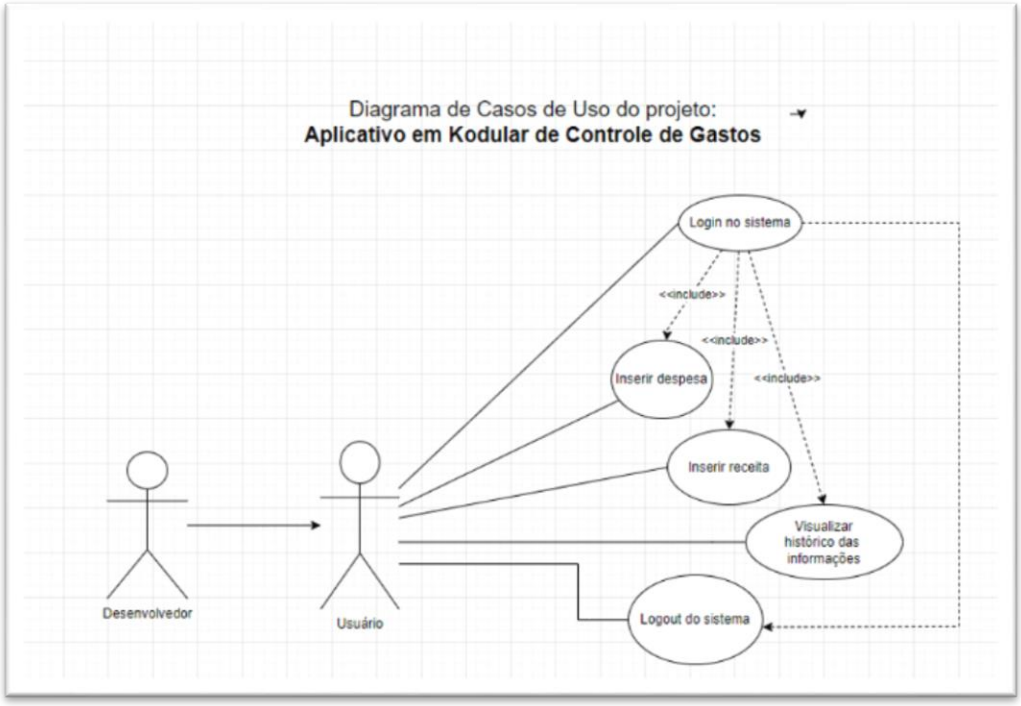
- data: Date: Armazena a data da receita.

- hora: Time: Armazena a hora da receita.
- categoria: String: Indica a categoria à qual a receita pertence.
- valor: double: Representa o valor da receita.

Essas duas classes, Despesa e Receita, são usadas para armazenar informações detalhadas sobre transações financeiras no aplicativo.

3.3. Revisão e descrição dos diagramas (caso de uso/ classe)

Imagem 6 – Diagrama de caso de uso



Fonte: Dos próprios autores, 2023

No caso, nossa equipe desenvolvedora irá proporcionar para o usuário a possibilidade de logar no sistema com um login e senha de acesso, onde ele encontrará uma interface com as opções para inserir despesa, receita e visualizar o histórico criado por esse usuário. E ao terminar a sua atividade, terá o botão para logout do sistema, assim mantendo seguro os dados desse usuário caso outra pessoa utilize o mesmo aparelho celular.

3.4. Desenvolvimento das Estruturas de Banco de Dados

Para desenvolver estruturas de banco de dados para armazenar dados das classes "Usuario", "Despesa" e "Receita", utilizaremos um banco de dados relacional, como o SQLite.

3.4.1 Tabela de Usuários:

CREATE TABLE Usuarios (id INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(255) NOT NULL, senha VARCHAR(255) NOT NULL, totalReceita DOUBLE, totalDespesa DOUBLE, saldo DOUBLE, on BOOLEAN);

Criamos uma tabela chamada "Usuarios" para armazenar informações sobre os usuários do aplicativo. Cada usuário terá um ID único, um nome, senha, os totais de receita e despesa, saldo e um indicador "on" para verificar se o usuário está logado.

3.4.2 Tabela de Despesas:

CREATE TABLE Despesas (id INT AUTO_INCREMENT PRIMARY KEY, usuario_id

INT, data DATE, hora TIME, categoria VARCHAR(255), valor DOUBLE, FOREIGN KEY (usuario_id) REFERENCES Usuarios(id));

A tabela "Despesas" armazenará informações sobre as despesas financeiras. Cada despesa terá um ID único, o ID do usuário que a registrou, data, hora, categoria e valor. Estabelecemos uma relação entre a tabela de Despesas e a tabela de Usuários usando a chave estrangeira "usuario_id".

3.4.3 Tabela de Receitas:

CREATE TABLE Receitas (id INT AUTO_INCREMENT PRIMARY KEY, usuario_id INT, data DATE, hora TIME, categoria VARCHAR(255), valor DOUBLE, FOREIGN KEY (usuario_id) REFERENCES Usuarios(id));

A tabela "Receitas" armazenará informações sobre as receitas financeiras. Ela é semelhante à tabela de Despesas, com campos para data, hora, categoria e valor. Também estabelecemos uma relação entre a tabela de Receitas e a tabela de Usuários usando a chave estrangeira "usuario_id".

Com essas estruturas de banco de dados, é possível armazenar e gerenciar os dados dos usuários, despesas e receitas no aplicativo de gerenciamento financeiro.

4. IMPLEMENTAÇÃO DO SISTEMA

4.1. Tela Inicial de Login 2.0.

Criação do designer da **TelaPrincipal**, aqui é onde ocorrerá a interação do usuário com o objetivo do aplicativo.

Foram criados os componentes:

TelaPrincipal	**TELA DE AÇÃO DO APLICATIVO**
Background Color	#444444FF
Align Horizontal	Center
Align Vertical	Center
Title Visible	Desabilitado
Vertical_Scroll_TelaPrincipal	**ORGANIZAR OS ELEMENTOS NA TELA INTEIRA VERTICALMENTE**
Align Horizontal	Center
Align Vertical	Center
Background Color	#00000000
Height	Fill Parent
Width	Fill Parent
Scrollbar	Habilitado
Visible	Habilitado
Horizontal_BotoesTelaPrincipal	**ORGANIZAR OS BOTÕES DE RECEITA E DESPESA NA TELA HORIZONTALMENTE**
Align Horizontal	Center
Align Vertical	Top
Background Color	#00000000
Height	Automático
Width	Fill Parent

Horizontal_BotoesTelaPrincipal	**ORGANIZAR OS BOTÕES DE RECEITA E DESPESA NA TELA HORIZONTALMENTE**
Visible	Habilitado
btn_despesa	**BOTÃO PARA INCLUIR DESPESA**
Background Color	#F44336FF
Enabled	Habilitado
Font Bold	Habilitado
Font Size	18px
Height	50px
Width	Fill parent
Shape	Rounded
Text	- DESPESA
Text Aligment	Center
Hint Color	#FFFFFFFF
spc_botoes	**ESPAÇO**
Height	50px
Width	15px
Visible	Habilitado
btn_receita	**BOTÃO PARA INCLUIR RECEITA**
Background Color	#4CAF50FF
Enabled	Habilitado
Font Bold	Habilitado

btn_receita	**BOTÃO PARA INCLUIR RECEITA**
Font Size	18px
Height	50px
Width	Fill parent
Shape	Rounded
Text	+ RECEITA
Text Aligment	Center
Hint Color	#FFFFFFFF
Card_Dados	**CARD PARA VISUALIZAÇÃO E INTERAÇÃO DO USUÁRIO COM OS DADOS DO APLICATIVO**
Align Horizontal	Center
Align Vertical	Top
Background Color	#FFFFFFFF
Content Padding Bottom	8
Content Padding Left	8
Content Padding Right	8
Content Padding Top	8
Corner Radius	10
Elevation	2
Height	Automático
Width	Fill Parent
Stroke Color	#1E9EECFE

Card_Dados	**CARD PARA VISUALIZAÇÃO E INTERAÇÃO DO USUÁRIO COM OS DADOS DO APLICATIVO**
Stroke Width	5
Visible	Habilitado
Horizontal_Dados	**ORGANIZAR AS LEGENDAS E OS VALORES NA TELA HORIZONTALMENTE**
Align Horizontal	Left
Align Vertical	Center
Background Color	#00000000
Height	Fill Parent
Width	Fill Parent
Scrollbar	Desabilitado
Visible	Habilitado
Vertical_Legendas	**ORGANIZAR AS LEGENDAS NA TELA VERTICALMENTE**
Align Horizontal	Center
Align Vertical	Top
Background Color	#00000000
Height	Automático
Width	Automático
Scrollbar	Desabilitado
Visible	Habilitado

lbl_receita	**LABEL COM A RECEITA DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	Receita:
Text Aligment	Center
Text Color	#4CAF50FF
Visible	Habilitado
lbl_despesa	**LABEL COM A DESPESA DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	Despesa:
Text Aligment	Center
Text Color	#F44336FF
Visible	Habilitado

lbl_total	**LABEL COM A TOTAL DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	Total:
Text Aligment	Center
Text Color	#1E9EECFE
Visible	Habilitado
spc_Dados	**ESPAÇO**
Height	Automático
Width	Fill Parent
Visible	Habilitado
Vertical_Valores	**ORGANIZAR OS VALORES NA TELA VERTICALMENTE**
Align Horizontal	Center
Align Vertical	Top
Background Color	#00000000
Height	Automático
Width	Automático
Scrollbar	Desabilitado

Vertical_Valores	**ORGANIZAR OS VALORES NA TELA VERTICALMENTE**
Visible	Habilitado
lbl_ValorReceita	**LABEL COM O VALOR DA RECEITA DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	
Text Aligment	Center
Text Color	#4CAF50FF
Visible	Habilitado
lbl_ValorDespesa	**LABEL COM O VALOR DA DESPESA DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	
Text Aligment	Center

lbl_ValorDespesa	**LABEL COM O VALOR DA DESPESA DO USUÁRIO**
Text Color	#F44336FF
Visible	Habilitado
lbl_ValorTotal	**LABEL COM O VALOR TOTAL DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	16px
Height	Automático
Width	Automático
Text	
Text Aligment	Center
Text Color	#1E9EECFE
Visible	Habilitado
lbl_UltimosRegistros	**LABEL COM O TÍTULO DA LISTA COM OS ÚLTIMOS REGISTROS DO USUÁRIO**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	20px
Height	Automático
Width	Automático

lbl_UltimosRegistros	**LABEL COM O TÍTULO DA LISTA COM OS ÚLTIMOS REGISTROS DO USUÁRIO**
Text	Últimos Registros
Text Aligment	Center
Text Color	#000000FF
Visible	Habilitado
list_Valores	**LISTA COM OS ÚLTIMOS REGISTROS DO USUÁRIO**
Background Color	#607D8BFF
Heigth	Automático
Width	Fill Parent
Item Height in %	10
Scrolling Fading	Habilitado
Scrolling Speed	1.0
Search Text Color Color	#FFFFFFFF
Show Scrollbar	Habilitado
Text Alignment	Center
Text Color	#000000FF
Font Size	18
Visible	Habilitado

Vertical_DadosFinanceiros	**ORGANIZAR OS DADOS FINANCEIROS NA TELA VERTICALMENTE**
Align Horizontal	Center
Align Vertical	Top
Background Color	#F5F1F1FE
Height	Automático
Width	Fill Parent
Visible	Habilitado
txt_valor	**CAIXA DE TEXTO COM OS VALORES COLOCADOS PELO USUÁRIO**
Background Color	#C5C0C0FE
Font Bold	Habilitado
Font Size	18px
Height	40px
Width	200px
Hint	Valor
Hint Color	#EEEEEEFF
Input type	Normal
Text Aligment	Center
Hint Color	#000000FF

Horizontal_DataHora	**ORGANIZAR OS DADOS DE DATA E HORA NA TELA HORIZONTALMENTE**
Align Horizontal	Center
Align Vertical	Center
Background Color	#00000000
Height	Fill Parent
Width	Fill Parent
Visible	Habilitado
lbl_hora	**LABEL COM O TEXTO DE REFERÊNCIA HORA**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	Horário:
Text Aligment	Left
Text Color	#000000FF
Visible	Habilitado
Hora	**COMPONENTE PARA VISUALIZAR DATA E HORA**
Background Color	#FFFFFF00

Hora	**COMPONENTE PARA VISUALIZAR DATA E HORA**
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	
Text Aligment	Center
Text Color	#000000FF
Visible	Habilitado
lbl_data	**LABEL COM O TEXTO DE REFERÊNCIA DATA**
Background Color	#FFFFFF00
Font Bold	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	Data:
Text Aligment	Left
Text Color	#000000FF
Visible	Habilitado

Data	**COMPONENTE PARA VISUALIZAR DATA E HORA**
Background Color	#FFFFFF00
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	
Text Aligment	Center
Text Color	#000000FF
Visible	Habilitado
Horizontal_Categoria	**ORGANIZAR OS DADOS DE CATEGORIA NA TELA HORIZONTALMENTE**
Align Horizontal	Center
Align Vertical	Center
Background Color	#00000000
Height	Automático
Width	Fill Parent
Visible	Habilitado
lbl_categoria	**LABEL COM O TEXTO DE REFERÊNCIA CATEGORIA**
Background Color	#FFFFFF00

lbl_categoria	**LABEL COM O TEXTO DE REFERÊNCIA CATEGORIA**
Font Bold	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	Categoria:
Text Aligment	Center
Text Color	#000000FF
Visible	Habilitado
list_categoria	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Background Color	#00000000
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	1px
Text	
Text Aligment	Center
Text Color	#FFFFFF
Visible	Habilitado

btn_adicionarValor	**BOTÃO PARA ENVIAR DADOS INSERIDOS PELO USUÁRIO**
Background Color	#1E9EECFE
Enabled	Habilitado
Font Bold	Habilitado
Font Size	16px
Height	40px
Width	200px
Shape	Rounded
Text	Ok
Text Aligment	Center
Hint Color	#FFFFFFFF
Visible	Habilitado
spc_rodade1	**ESPAÇO**
Height	40px
Width	Automático
Visible	Habilitado
btn_Voltar	**BOTÃO PARA RETORNAR PARA SCREEN1, TELA DE LOGIN**
Background Color	#000000FF
Enabled	Habilitado
Font Bold	Habilitado
Font Size	18px

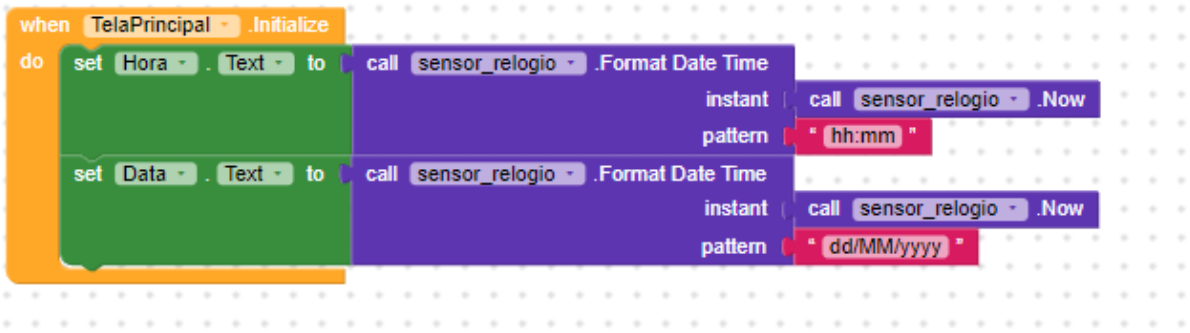
btn_Voltar	**BOTÃO PARA RETORNAR PARA SCREEN1, TELA DE LOGIN**
Height	40px
Width	200px
Shape	Rounded
Text	Voltar
Text Aligment	Center
Hint Color	#FFFFFFFF
Visible	Habilitado
spc_rodade2	**ESPAÇO**
Height	40px
Width	Automático
Visible	Habilitado

Não são visíveis na tela

sensor_relogio	**SENSOR QUE PEGA A DATA E HORA DO APARELHO**
Timer Always Fires	Desabilitado
Timer Enabled	Desabilitado
Timer Interval	1000

Desenvolvimento da lógica de programação dos blocos

Imagem 7 – Bloco de atualização do horário



Fonte: Dos próprios autores, 2023.

Dentro desse bloco chamamos o **sensor_relogio** e formatamos a data e a hora para o formato brasileiro. Assim que a **TelaPrincipal** for inicializada isso ocorrerá e atualizará o horário e a data do telefone móvel.

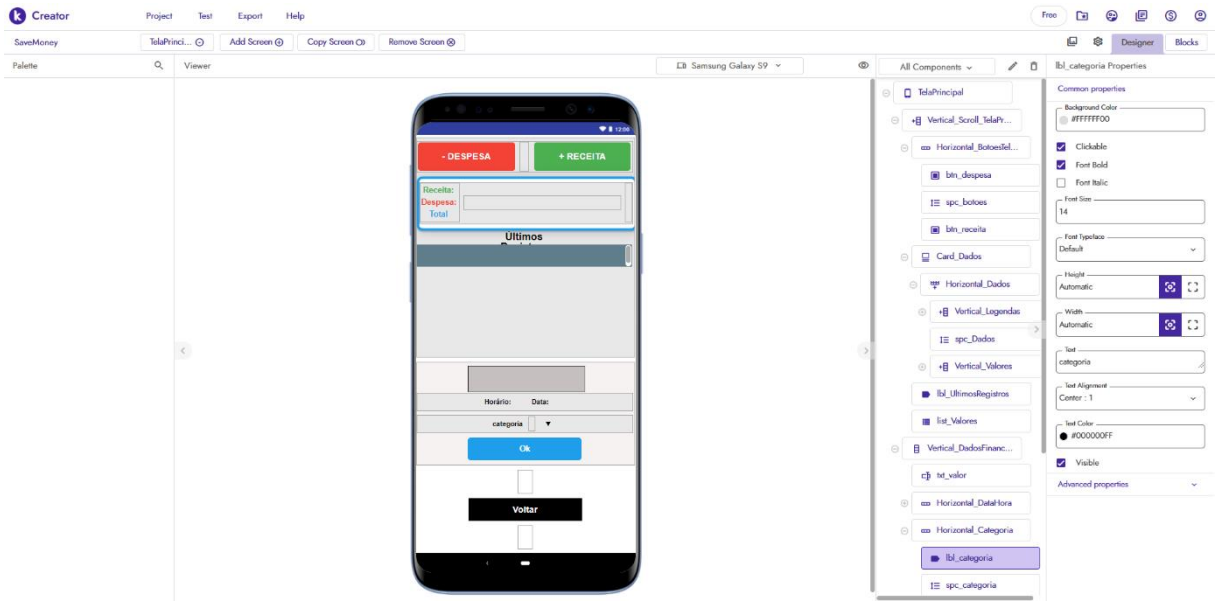
Imagem 8 – Bloco de direcionamento ao login

Fonte: Dos próprios autores, 2023.

Ao clicar no botão o usuário será encaminhado para a página **Screen1** onde está o login.

4.2. Tela Principal 2.1.

Imagem 9 – Implementação do design da tela inicial



Fonte: Dos próprios autores, 2023.

Nessa parte desenvolvemos os ajustes de hora, data e categoria no btn_despesa.

Os componentes incluídos e modificados:

Horizontal_Categoria	**ORGANIZAR OS DADOS DE CATEGORIA NA TELA HORIZONTALMENTE**
Align Horizontal	Center
Align Vertical	Center
Background Color	#00000000

Horizontal_Categoria	**ORGANIZAR OS DADOS DE CATEGORIA NA TELA HORIZONTALMENTE**
Clickable	Habilitado
Height	Automático
Width	Fill Parent
Visible	Desabilitado
lbl_categoria	**LABEL COM O TEXTO DE REFERÊNCIA CATEGORIA**
Background Color	#FFFFFF00
Clickable	Habilitado
Font Bold	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Text	categoria
Text Aligment	Center
Text Color	#000000FF
Visible	Habilitado
spc_categoria	**ESPAÇO**
Height	Automático
Width	10px
Visible	Habilitado

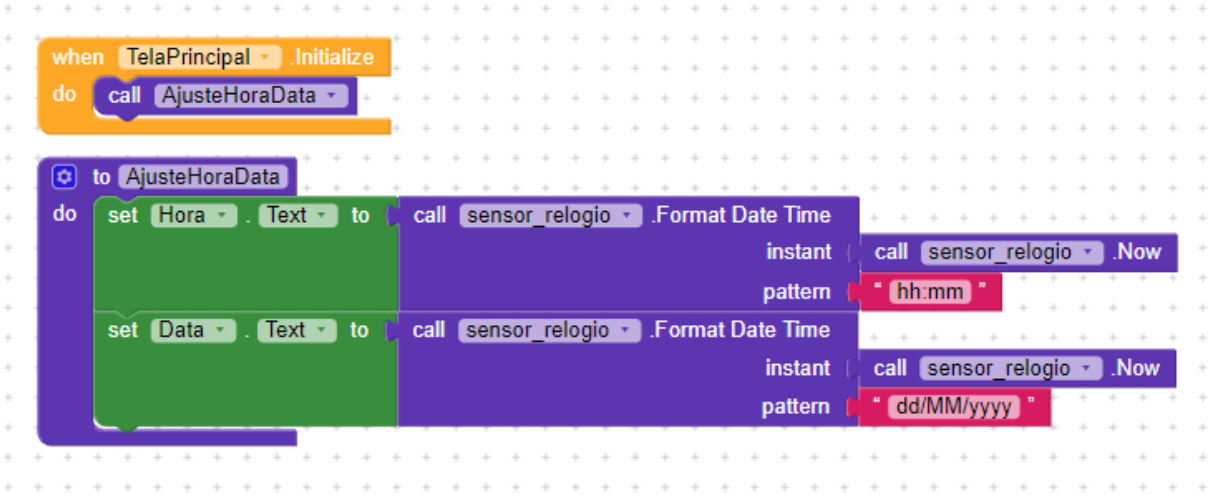
list_categoria	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Background Color	#FFFFFF00
Elements from String	Casa, Trabalho, Entretenimento, Transporte, Banco
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	Automático
Selection	Categoria
Text	▼
Text Aligment	Center
Text Color	#FFFFFF
Visible	Habilitado

Não são vísiveis na tela

notifica	**COMPONENTE RESPONSÁVEL POR NOTIFICAÇÕES PARA O USUÁRIO**
	Não foi modificado nada neste componente

Desenvolvimento da lógica de programação dos blocos

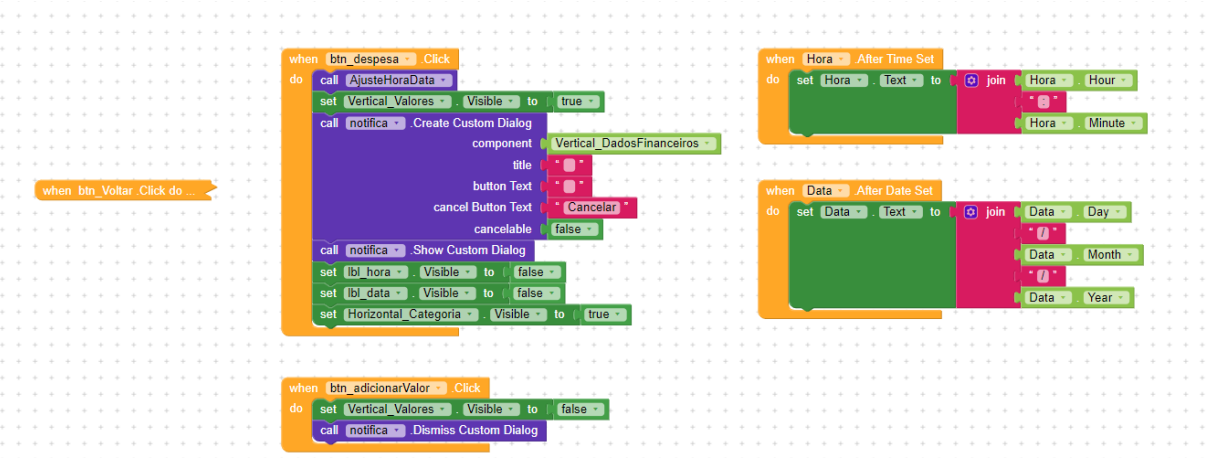
Imagem 10 – bloco AjusteHoraData



Fonte: Dos próprios autores, 2023.

Criamos aqui um procedimento de ação, que quando precisar atualizar o horário e a data para ficar igual ao do aparelho, iremos chamar essa ação. Substituimos os blocos que foram inseridos inicialmente no inicializar **TelaPrincipal** para a função **AjusteHoraData**.

Imagem 11 – Bloco de funcionalidades do botão despesa



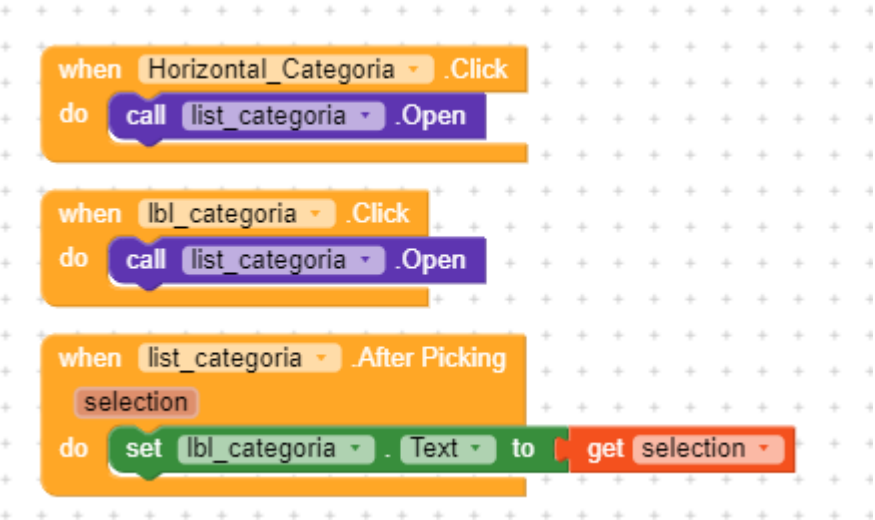
Fonte: Dos próprios autores, 2023.

Aqui inserimos as primeiras funcionalidades do botão **btn_despesa**. Quando ele for acionado ele irá chamar o elemento de notificação, que mostrará o bloco de componentes **Vertical_DadosFinanceiros**, onde foi optado por deixar o texto **Cancelar** para caso o usuário queira voltar sem salvar nada e as outras opções foram desligadas e deixadas vazias. Nesse espaço aparecerá a o campo para inserir o valor, a hora, a data e a categoria do valor, sendo que desativamos as legendas **data** e **hora** por ser intuitivo e um visual mais limpo para o usuário.

Quando o usuário sair da tela de visualizar a inserção da despesa, clicando no **btn_adicionarValor** e sendo enviado de volta para a tela inicial ele não verá mais a opção **Vertical_Valores**.

A hora e a data poderão ser selecionadas pelo usuário, por isso os blocos estão chamando a **Data** e **Hora** e escrevendo o valor que o usuário escolher por cima do valor anterior.

Imagem 12 – Bloco de listagem



Fonte: Dos próprios autores, 2023.

Já nesses últimos blocos, quando o **Horizontal_Categoria** e o texto do **lbl_categoria** forem clicados ele abrirá a lista de categorias.

Essa lista pegará o valor selecionado e escreverá ele para visualização do

usuário no **lbl_categoria**.

4.3. Tela Principal 2.2.

Desenvolvimento do Botão Despesa

Desenvolvimento de salvar valores do botão **Despesa** e apresentar a lista para o usuário.

Os componentes incluídos e modificados:

list_categoria	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Elements from String	Categoria, Casa, Trabalho, Entretenimento, Transporte, Banco
btn_Voltar	**COMPONENTE RESPONSÁVEL PELA SAÍDA DA TELA PRINCIPAL PARA A TELA SCREEN1**
Text	Logout
list_Valores	**LISTA COM OS ÚLTIMOS REGISTROS DO USUÁRIO**
Item Height in %	5
Text Alignment	Left
Font Size	14

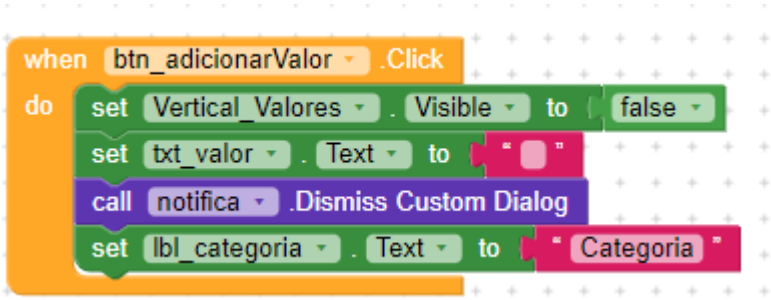
Mudança do texto para ser mais intuitivo para o usuário sair da aplicação.

Não são visíveis na tela

Tiny_DB1	**COMPONENTE RESPONSÁVEL PELO BANCO DE DADOS**
	Não foi modificado nada neste componente

Desenvolvimento da lógica de programação dos blocos

Imagem 13 – funcionalidade do botão adicionar valor



Fonte: Dos próprios autores, 2023.

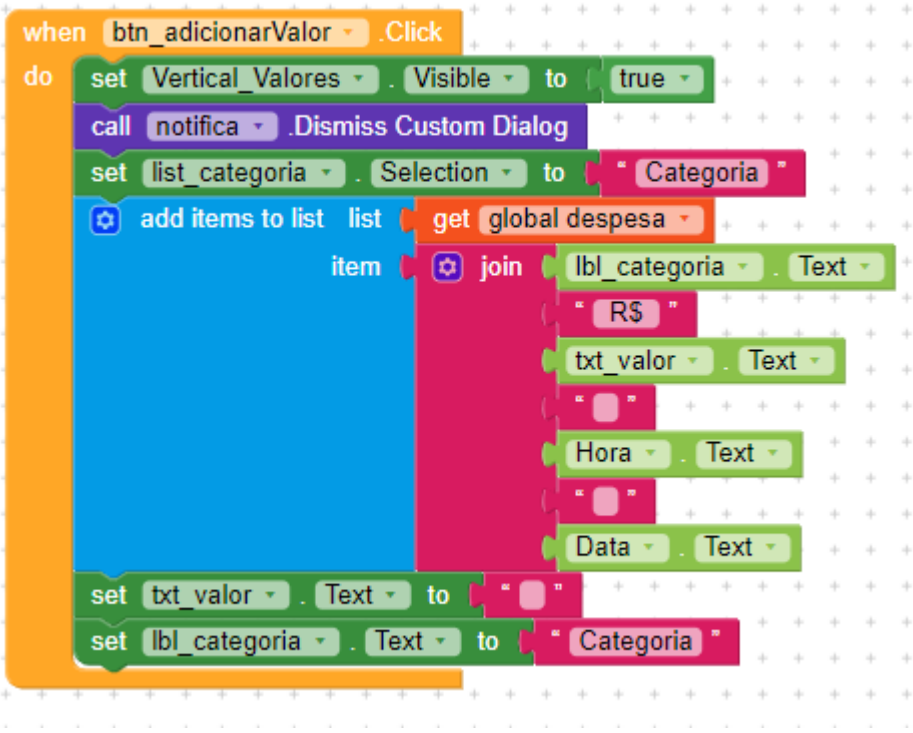
Aqui inserimos o espaço vazio e a opção **Categoria** para que o usuário sempre quando entrar na despesa poder preencher novamente o que deseja, sem ficar os dados anteriores aparecendo. Optamos por substituir o texto da categoria selecionado por **Categoria** sempre que o **btn_adicionarValor** for clicado.

Imagem 14 – Criação da lista de despesas

Fonte: Dos próprios autores, 2023.

Criação da variável chamada **despesa** que cria uma lista vazia.

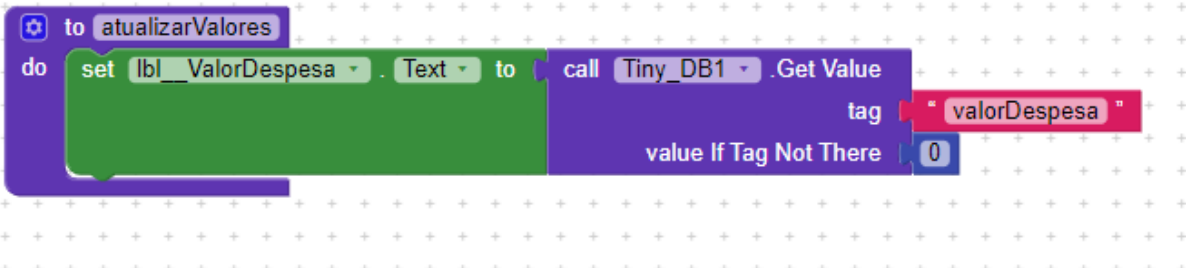
Imagem 15 – Funcionalidade do botão adicionar valor



Fonte: Dos próprios autores, 2023.

Agora vamos pegar o componente de adicionar itens a lista, que colocará o valor da variável **despesa** que o usuário inseriu no **txt_Valor** e mostrará a categoria, o R$ valor, hora e data que estão armazenados no **Tiny_DB1**. Os set de **txt_valor** e **lbl_categoria** estão por último para não excluir os valores antes de salvar, quando tudo for incluído no banco, depois apaga tudo e retorna limpo para o usuário inserir os dados novamente.

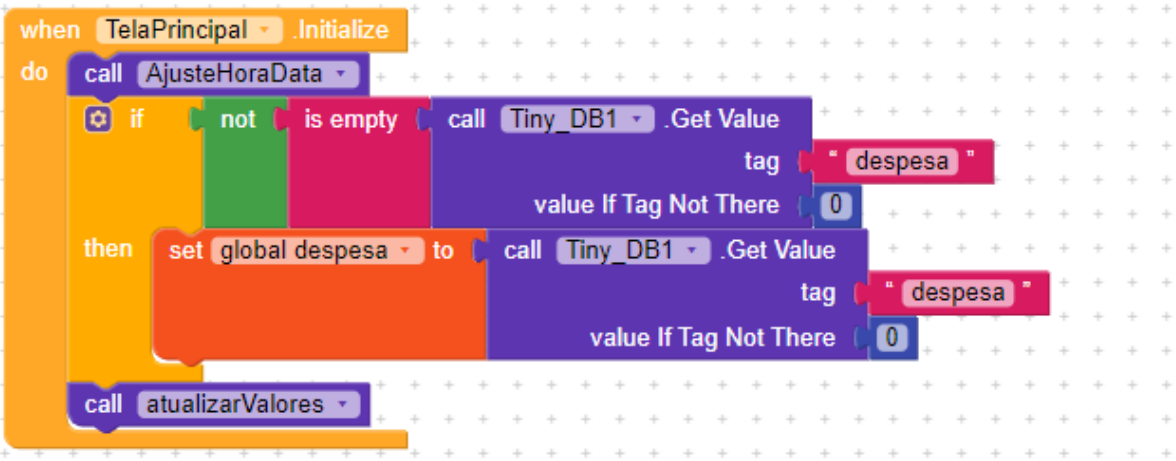
Imagem 16 – Função de atualização de valores



Fonte: Dos próprios autores, 2023.

Criamos uma função com o nome **atualizarValores** que vai ser sempre chamada quando iniciar a tela principal. Ela pega o valor do banco de dados e coloca a função no **lbl__ValorDespesa**.

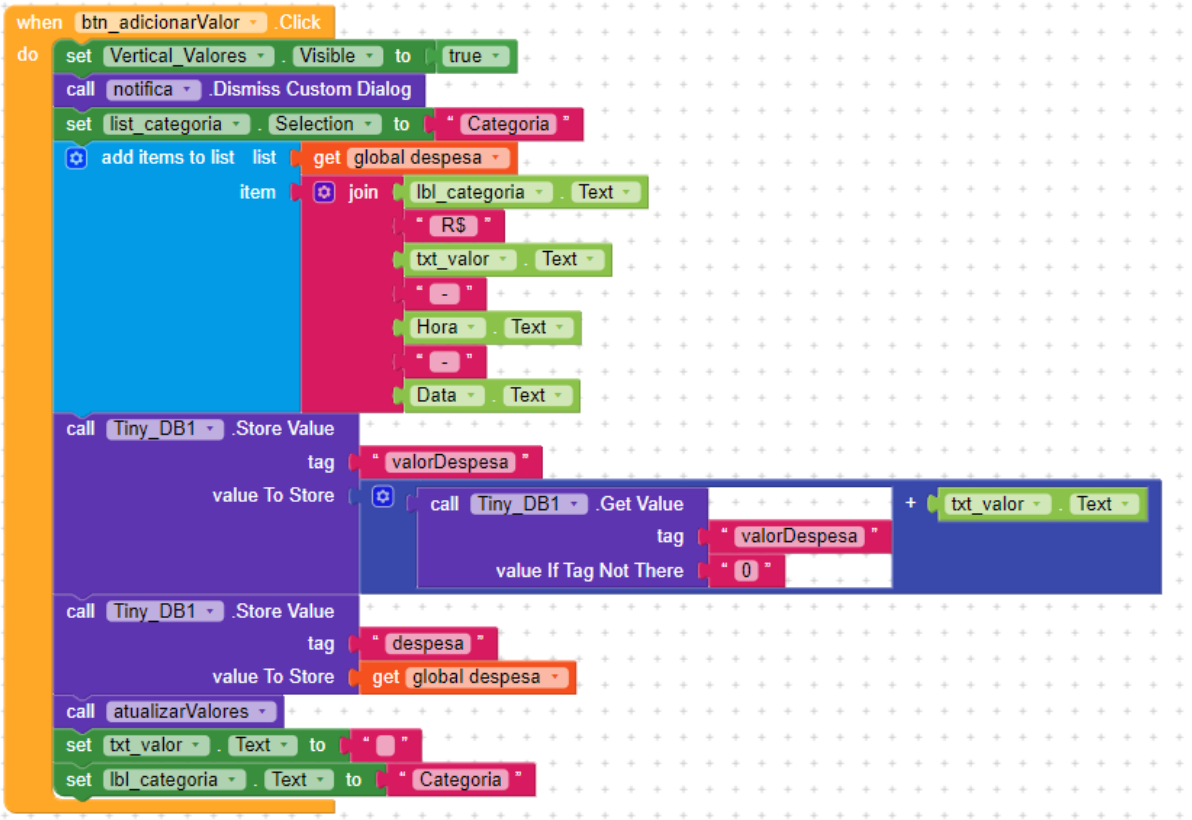
Imagem 17 – Bloco de inicialização da tela inicial



Fonte: Dos próprios autores, 2023.

Dentro do bloco **TelaPrincipal** irá fazer uma verificação que se não estiver vazio o nosso banco de dados ele irá ajustar a variável **global despesa** e por último chamar a função **atualizarValores**.

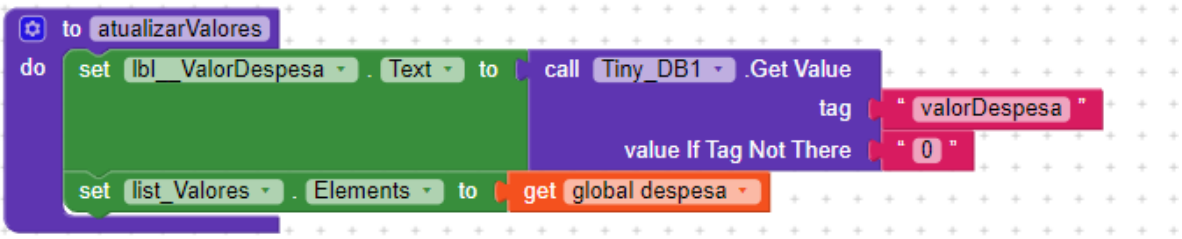
Imagem 18 – Funcionalidade do botão adicionar valor



Fonte: Dos próprios autores, 2023.

Aqui já incluimos o bloco do banco de dados que chamará os valores e somará com o **txt_valor** inserido pelo usuário. Usamos outro nome para o banco de dado armazenar **valorDespesa** separado do outro valor guardado **despesa**.

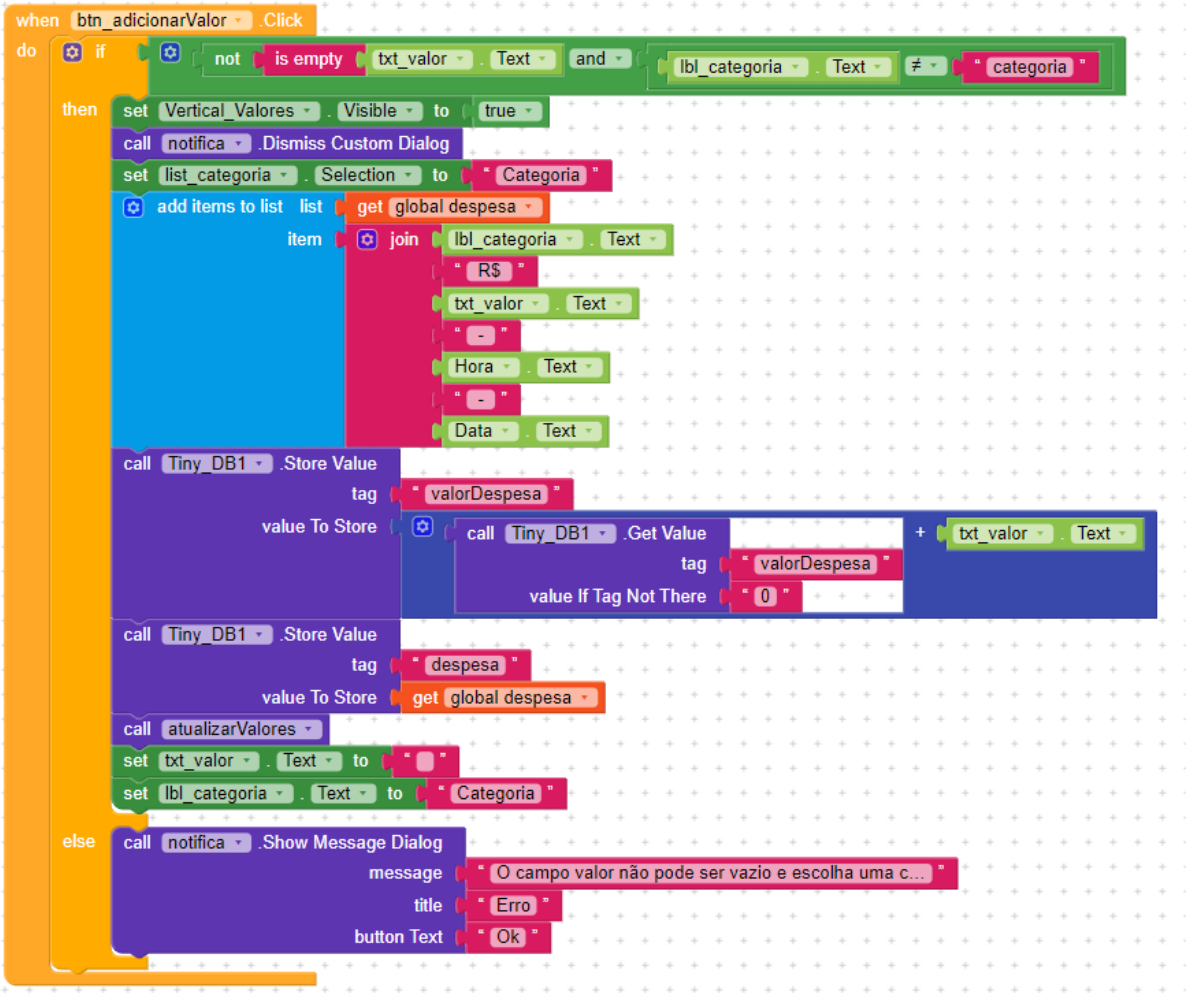
Imagem 19 – Função atualiza valores



Fonte: Dos próprios autores, 2023.

Colocamos os valores para testar da **global despesa** na **list_Valores**. Obs.: Isso é apenas provisório, pois nesse bloco apresentaremos a receita e a despesa. Isso é apenas um teste.

Imagem 20 – Funcionalidade do botão adicionar valor

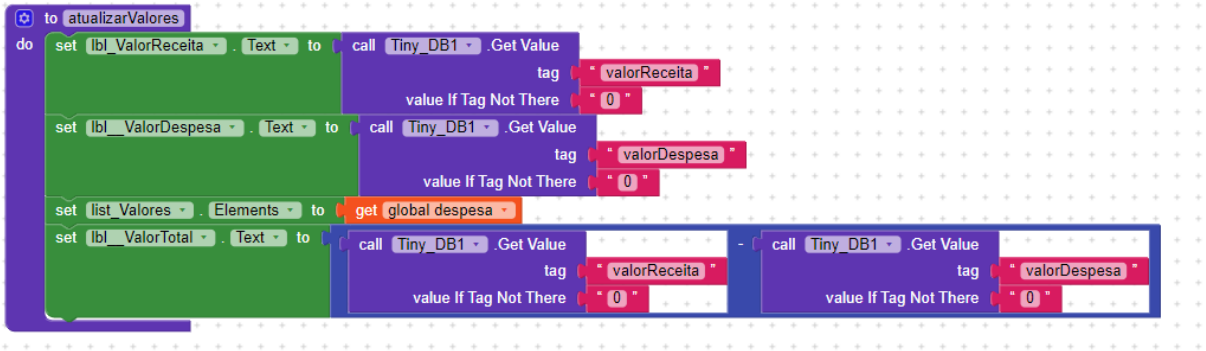


Fonte: Dos próprios autores, 2023.

Para finalizar esse bloco por enquanto colocamos uma condição para verificar se o campo **txt_valor** está vazio e se a categoria foi selecionada, caso a **lbl_categoria** seja diferente de **categoria** significa que foi selecionada. Caso os campos não estejam correspondentes a condição uma mensagem de erro será apresentada ao usuário. A mensagem de erro que pode aparecer sem essa

condicional está no final desse documento.

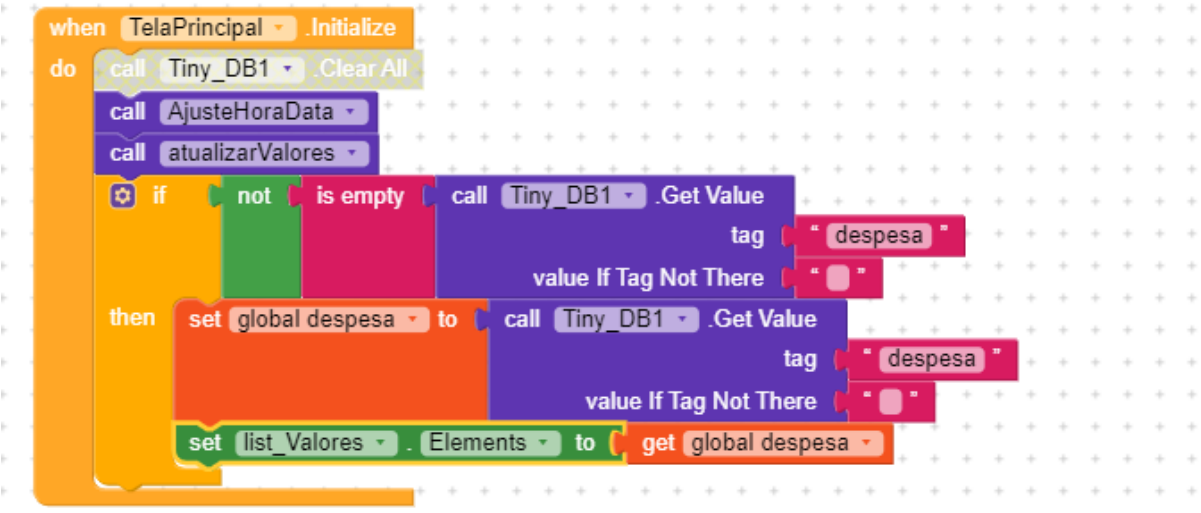
Imagem 21 – Função atualizarValores



Fonte: Dos próprios autores, 2023.

Aqui já atualizamos os valores da **Horizontal_Dados**, onde o valor total é a receita menos a despesa e a receita como ainda não foi feita está com o valor 0.

Imagem 22 – Funcionalidades da Tela Principal ao ser inicializada



Fonte: Dos próprios autores, 2023.

E agora por fim, ao inicializar a tela principal será chamado os componentes do **list_Valores**, o procedimento que atualizará os dados **atualizarValores** e os dados do banco de dados.

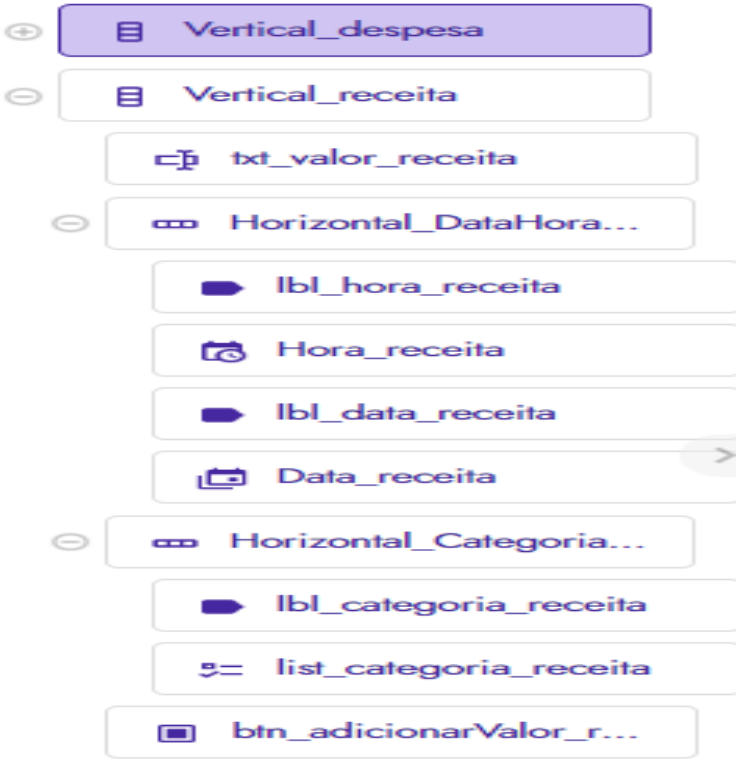
4.4. Tela Principal 2.3

Desenvolvimento do botão receita

Iremos desenvolver nesse bloco a inclusão da funcionalidade do btn_receita e alterar algumas nomenclaturas.

Foram criados e modificados os componentes:

Imagem 23 – Componentes criados na tela inicial



Fonte: Dos próprios autores, 2023.

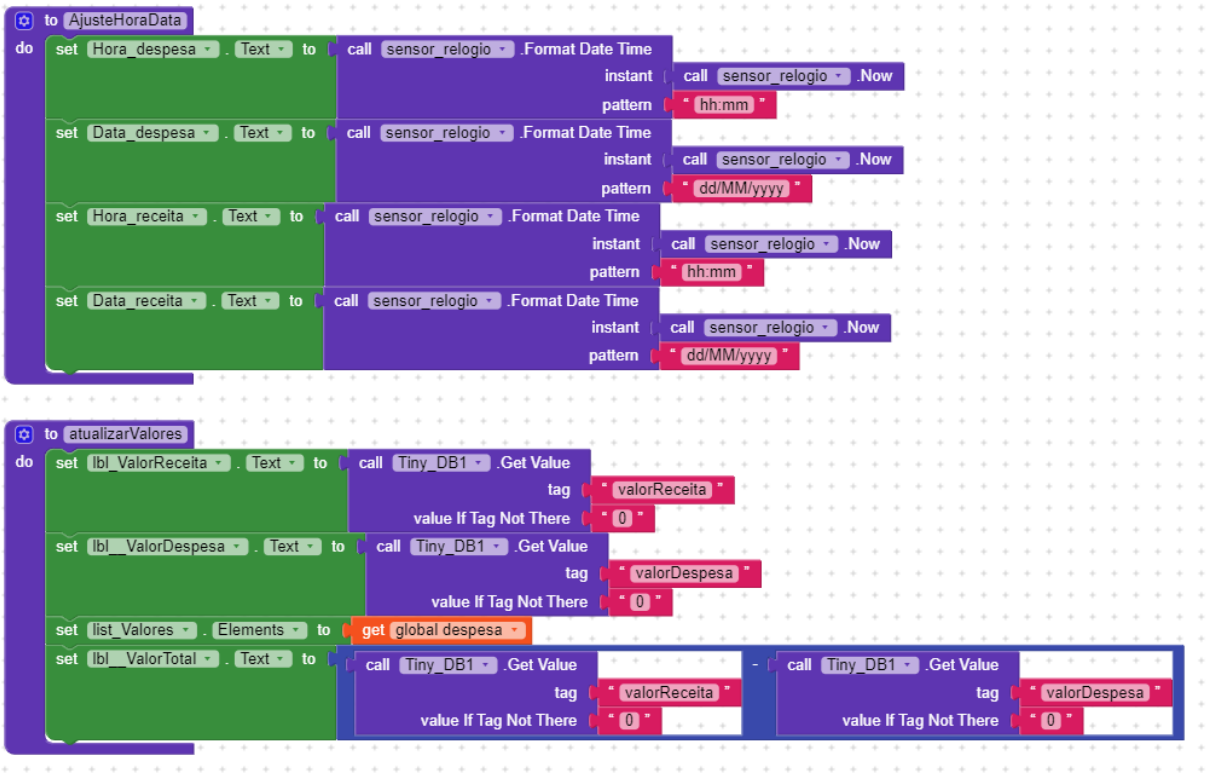
Nesse caso fizemos uma duplicada do **Vertical_despesa** e modificamos a nomenclatura para não ocasionar erros na hora de selecionar o componente de cada um.

list_categoria_despesa	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Background Color	#00000000
Elements from String	Categoria, Casa, Trabalho, Entretenimento, Transporte, Banco
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	1px
Text	
Text Aligment	Center
Text Color	#FFFFFF
Visible	Habilitado
list_categoria_receita	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Background Color	#00000000
Elements from String	Categoria, Salário, 13º , Extra
Enabled	Habilitado
Font Size	14px
Height	Automático
Width	1px
Text	

list_categoria_receita	**LISTA DE ITEMS COM AS CATEGORIAS DESPESA OU RECEITA PARA O VALOR**
Text Aligment	Center
Text Color	#FFFFFF
Visible	Habilitado

Desenvolvimento da lógica de programação dos blocos

Imagem 24 – Funcionalidade AjusteHoraData e atualizarValores



Fonte: Dos próprios autores, 2023.

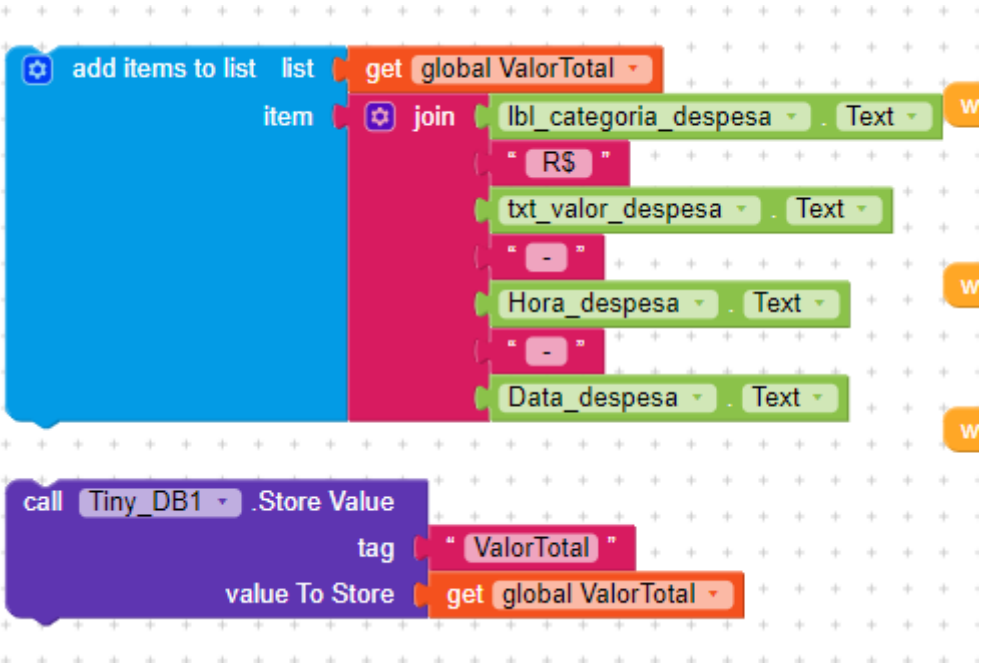
Aqui incluímos os blocos da receita junto com o da despesa.

Imagem 25 – Criação da lista vazia

Fonte: Dos próprios autores, 2023.

Criamos uma nova lista para receber os valores de receita e despesa e apresentar para o usuário.

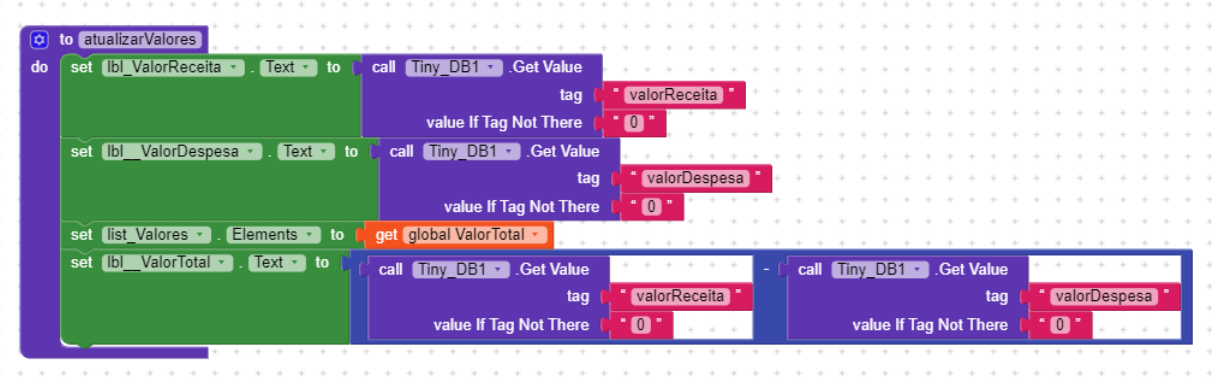
Imagem 26 – Blocos que adicionam itens à lista



Fonte: Dos próprios autores, 2023.

Criamos esses blocos para inserir na despesa e na receita.

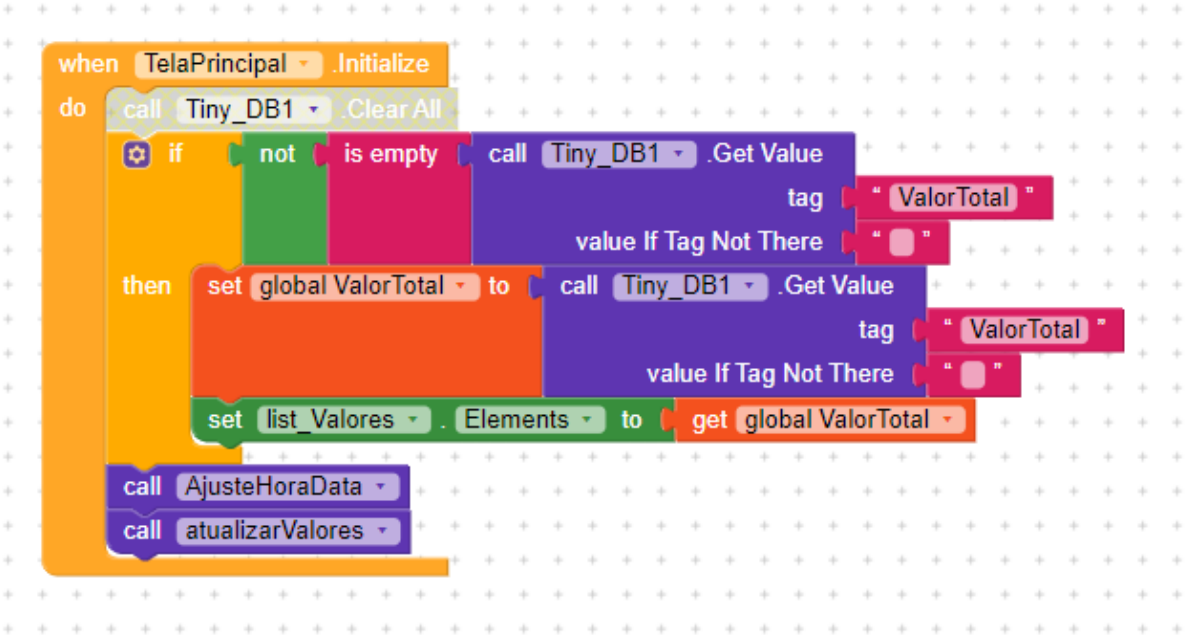
Imagem 27 – Atualizando Valores



Fonte: Dos próprios autores, 2023.

Depois chamamos no **list_Valores**.

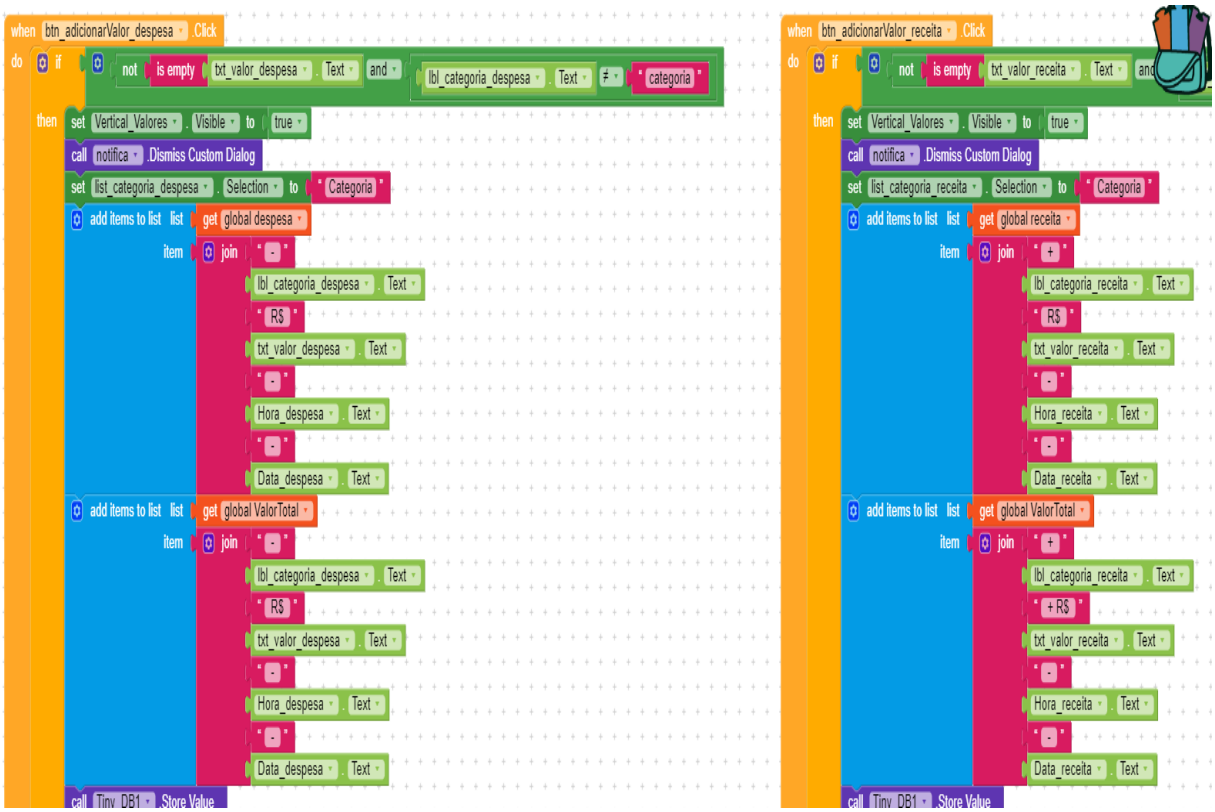
Imagem 28 – Inicialização da tela principal atualizada



Fonte: Dos próprios autores, 2023.

Colocamos o **ValorTotal** para aparecer na tela a receita e a despesa para o usuário. O bloco **ClearAll** está desabilitado pois ele só estava sendo usado para testar o programa.

Imagem 29 – Distinção entre Despesa e Receita

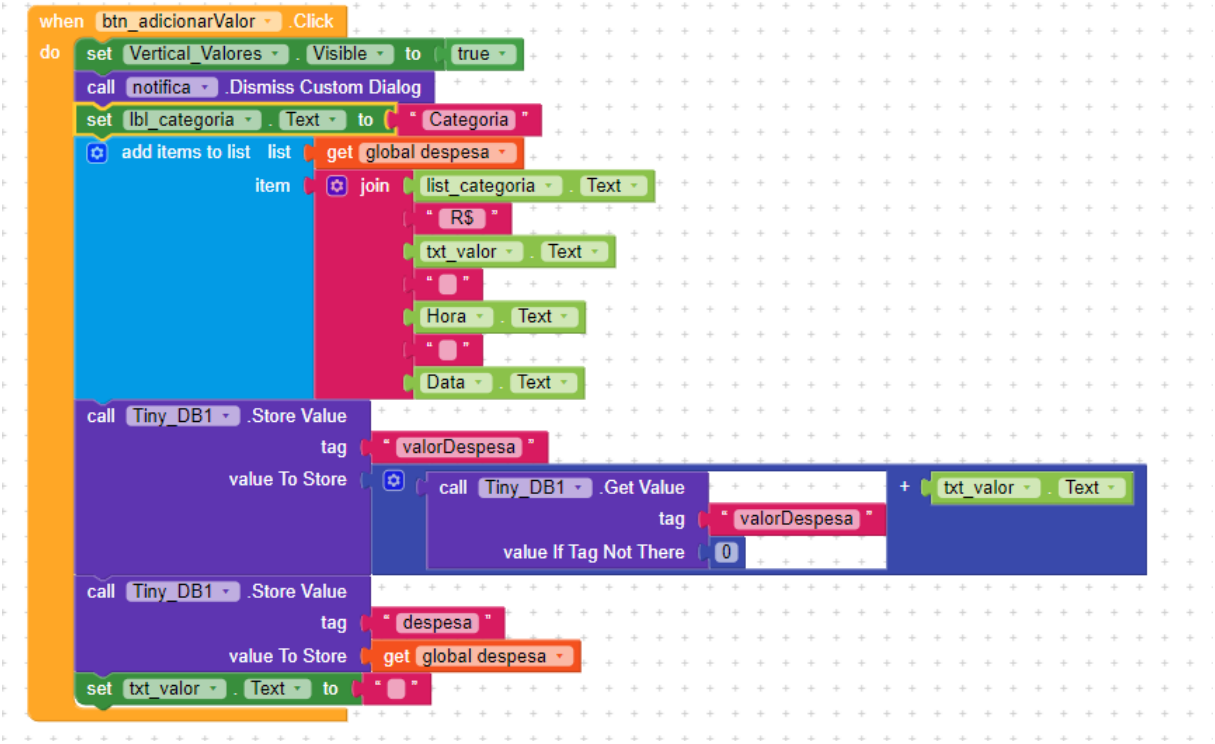


Fonte: Dos próprios autores, 2023.

E para mostrar o que é despesa e o que é receita nesse primeiro momento inserimos um + e um - no texto.

5. ROTEIRO DE TESTES E ERROS APRESENTADOS

Imagem 30 – Mensagens de erro da tela principal



Fonte: Dos próprios autores, 2023.

Alteramos o **txt_valor** para que ele não pegasse o resultado em branco nos retornando o erro:

The operation + cannot accept the arguments, [0], [empty-string].

Que significa que a operação não pode aceitar a soma de um texto vazio com o zero, pois o que foi colocado pelo usuário foi deletado antes de salvar.

6. CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, foi possível desenvolver e analisar o aplicativo "Save Money" no ambiente Kodular, com o objetivo principal de fornecer aos usuários uma ferramenta simples e eficiente para o controle de gastos e receitas. O projeto foi concebido como parte do trabalho de conclusão de curso de desenvolvimento de sistemas, proporcionando uma oportunidade valiosa para aplicar os conhecimentos adquiridos ao longo da formação acadêmica.

Durante o desenvolvimento do aplicativo, exploramos diversas funcionalidades proporcionadas pela plataforma Kodular, buscando uma abordagem intuitiva e amigável para os usuários. A capacidade de registrar gastos e receitas de forma fácil e rápida foi o cerne do projeto, visando atender a demanda crescente por soluções que auxiliem na organização financeira pessoal.

Principais Conclusões e Contribuições:

Facilidade de Uso: O aplicativo desenvolvido demonstrou uma interface intuitiva, permitindo que usuários, independentemente de seu nível de habilidade tecnológica, pudessem registrar suas transações financeiras de maneira descomplicada.

Controle Financeiro Pessoal: A implementação do controle de gastos e receitas proporcionou aos usuários uma ferramenta eficaz para monitorar suas finanças, promovendo uma conscientização sobre os padrões de consumo e contribuindo para uma gestão mais eficiente dos recursos financeiros.

Total de Receitas, Gastos e Saldo Líquido: A apresentação clara e organizada do total de receitas, gastos e saldo líquido oferece aos usuários uma visão abrangente de sua situação financeira, facilitando a tomada de decisões informadas.

Aplicação Prática do Conhecimento Adquirido: O desenvolvimento do aplicativo no ambiente Kodular permitiu a aplicação prática dos conhecimentos teóricos adquiridos ao longo do curso, consolidando habilidades de programação, design de interface e resolução de problemas.

Recomendações para Ações Futuras e Melhorias:

Implementação de Recursos Avançados: Para futuras versões, considera-se a implementação de recursos avançados, como a geração de relatórios gráficos, categorização de despesas e a integração com serviços bancários para automatizar a entrada de transações.

Aprimoramento da Segurança: Avaliar a inclusão de medidas adicionais de segurança para proteger as informações financeiras dos usuários, como autenticação de dois fatores e criptografia de dados sensíveis.

Feedback dos Usuários: Coletar feedback dos usuários para identificar pontos de melhoria e novas funcionalidades desejadas, promovendo uma abordagem orientada pelo usuário na evolução do aplicativo.

Expansão para Outras Plataformas: Considerar a expansão do aplicativo para outras plataformas além do Kodular, ampliando o alcance e a acessibilidade do Save Money.

Em suma, o desenvolvimento do aplicativo "Save Money" representa uma contribuição tangível para o campo do desenvolvimento de sistemas e aplicativos móveis, proporcionando uma solução prática para o desafio comum do controle financeiro pessoal. As recomendações apresentadas visam aprimorar ainda mais a utilidade e eficácia do aplicativo, destacando a importância contínua do aprimoramento tecnológico no cenário atual.

REFERÊNCIAS

GIROLLI, Guilherme Henrique. **Desenvolvimento de Sistemas III - Agenda 01 - Trabalhando com áudio e utilizando o atributo OnClick no button**. GEEaD - Grupo de Estudos de Educação a Distância Centro de Educação Tecnológica Paula Souza. São Paulo, 2020.

GIROLLI, Guilherme Henrique. **Desenvolvimento de Sistemas III - Agenda 03 - Manipulação de Banco de Dados Local no Dispositivo**. GEEaD - Grupo de Estudos de Educação a Distância Centro de Educação Tecnológica Paula Souza. São Paulo, 2020.

GIROLLI, Guilherme Henrique. **Desenvolvimento de Sistemas III - Agenda 05 - Permissões**. GEEaD - Grupo de Estudos de Educação a Distância Centro de Educação Tecnológica Paula Souza. São Paulo, 2020.

Sem autor. **Get to know us better**. Kodular, 2023. Disponível em: <https://www.kodular.io/about/>. Acesso em: 15 de setembro de 2023.

Sem autor. **Documentation**. Sqlite.org, 2023. Disponível em: <https://www. https://www.sqlite.org/docs.html/> Acesso em: 15 de setembro de 2023.

ANEXOS

Link para o designer do figma: <https://www.figma.com/file/d9xMHGXBBI1vr52GlRgJAj/TCC-SaveMoney?type=design&node-id=0%3A1&mode=design&t=NNfj6ZYP0bEXh8gA-1>.

Link para o repositório no Github: < https://github.com/claudiadejesusdantas/AppSaveMoney>