**Nome do Projeto: Save My Money – Gestão financeira**

# Histórico de Revisões

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 04/03/2020 | 1.0 | Feito casos de uso, regras de negócio e matrizes | Debora |
| 19/03/2020 | 3.0 | Feito modelos do banco de dados, referencias, capa, contracapa e índice  analítico | Debora |

**1. Introdução**

# 1.1 Finalidade

Este documento especifica os requisitos do sistema My Books, fornece aos desenvolvedores informações necessárias para o projeto e implementação e também como o sistema deve se comportar informando os testes que devem ser efetuados.

## 1.2 Práticas sociais e ambientais

O projeto tem por finalidadeatender a necessidade das pessoas de gerenciar seus gastos e ter controle sobre suas contas, e adquirir habilidades em sanar dividas e/ou adquirir bens.

# 3. Modelo de casos de uso

**4. Especificações de casos de uso**

**UC01 - Cadastrar Despesas**

**1. Caso de uso: Cadastrar Despesas na aplicação**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de despesas na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Uma nova despesa cadastrada na aplicação

# 6. Fluxo Principal

1. O ator acessa a opção incluir uma despesa
2. O ator insere os dados obrigatórios para o cadastro da despesa
3. O ator clica na opção salvar
4. A aplicação exibe uma mensagem rápida: “Despesa cadastrada com sucesso”
5. A aplicação fecha a tela de cadastro e entra na tela inicial
6. Fim do caso de uso
7. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

**FE01-** Campos obrigatórios sem preenchimento

Este fluxo de erro tem a finalidade de apresentar ao usuário o cenário de erro na tentativa de cadastrar uma nova despesa sem o preenchimento dos campos obrigatórios

1. A aplicação não finaliza o cadastro e apresenta a mensagem “Valor deve ser maior que 0“.

**UC02 - Cadastrar Receitas**

**1. Caso de uso: Cadastrar Receitas na aplicação**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de receitas na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Uma nova receita cadastrada na aplicação

# 6. Fluxo Principal

1. O ator acessa a opção incluir uma receita
2. O ator insere os dados obrigatórios para o cadastro da receita
3. O ator clica na opção salvar
4. A aplicação exibe uma mensagem rápida: “Receita cadastrada com sucesso”
5. A aplicação fecha a tela de cadastro e entra na tela inicial
6. Fim do caso de uso
7. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

**FE01-** Campos obrigatórios sem preenchimento

Este fluxo de erro tem a finalidade de apresentar ao usuário o cenário de erro na tentativa de cadastrar uma nova receita sem o preenchimento dos campos obrigatórios

1. A aplicação não finaliza o cadastro e apresenta a mensagem “Valor deve ser maior que 0“.

**UC03 - Cadastrar Objetivos**

**1. Caso de uso: Cadastrar Objetivos na aplicação**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de objetivos na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Um novo objetivo cadastrado na aplicação

# 6. Fluxo Principal

1. O ator acessa a opção incluir um novo objetivo
2. O ator insere os dados obrigatórios para o cadastro de objetivo
3. O ator clica na opção salvar
4. A aplicação exibe uma mensagem rápida: “Objetivo cadastrado com sucesso”
5. A aplicação fecha a tela de cadastro e entra na tela inicial
6. Fim do caso de uso
7. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

**FE01-** Campos obrigatórios sem preenchimento

Este fluxo de erro tem a finalidade de apresentar ao usuário o cenário de erro na tentativa de cadastrar um novo objetivo sem o preenchimento dos campos obrigatórios

1. A aplicação não finaliza o cadastro e apresenta a mensagem “Valor deve ser maior que 0“.

**UC04 - Cadastrar Pagamento de Despesa**

**1. Caso de uso: Cadastrar Pagamentos na aplicação**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de pagamentos na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Uma despesa paga na aplicação

# 6. Fluxo Principal

1. O ator acessa a tela inicial e seleciona uma despesa
2. A aplicação abre a despesa em detalhes com os campos: Valor total, data de vencimento e de criação desabilitados
3. O ator seleciona a opção pago e inclui uma data de pagamento se for diferente da data atual
4. O ator clica na opção salvar
5. A aplicação exibe uma mensagem rápida: “Pagamento registrado com sucesso”
6. A aplicação fecha a tela de detalhes e abre a tela inicial
7. Fim do caso de uso
8. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

Não há

**UC05 - Cadastrar Recebimento de receitas**

**1. Caso de uso: Cadastrar Recebimentos na aplicação**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de recebimento na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Uma receita recebida na aplicação

# 6. Fluxo Principal

1. O ator acessa a tela inicial e seleciona a opção de receita
2. A aplicação abre uma tela com uma lista de receitas
3. O ator seleciona uma das receitas ainda não recebida
4. A aplicação abre a receita em detalhes com os campos: Valor total e de criação desabilitados
5. O ator seleciona a opção recebido e inclui uma data de recebimento se for diferente da data atual
6. O ator clica na opção salvar
7. A aplicação exibe uma mensagem rápida: “Receita registrada no saldo de contas com sucesso”
8. A aplicação fecha a tela de detalhes e abre a tela inicial
9. Fim do caso de uso
10. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

Não há

**UC05 – Consultar saldos**

**1. Caso de uso: Consultar saldos**

# 2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de consulta de saldos na aplicação

1. **Ator (es):**

Usuário do sistema

1. **Pré Condições**

Ter efetuado login com sucesso na aplicação

1. **Pós Condições (Resultados)**

Apresentar saldos de contas, receitas e despesas

# 6. Fluxo Principal

1. O ator acessa a tela inicial
2. A aplicação apresenta todos os saldos de contas, de receitas e depesas
3. Fim do caso de uso
4. **Fluxo Alternativo**

Não há

# 8. Fluxo de Exceção

Não há

# Regras de negocio

|  |  |
| --- | --- |
| **Regras de negocio** | **Definição** |
| **[RN01] -** Efetuar login | O usuário deve estar previamente cadastrado com nome de usuário e senha |
| **[RN02] -** Cadastrar despesa na aplicação | Ao cadastrar uma despesa e defini-la como paga, a aplicação deve diminuir o valor do saldo de contas com o valor da despesa paga, como na formula abaixo:  saldoConta = saldoConta - ValorDepesa |
| **[RN02] -** Cadastrar despesa na aplicação | Uma despesa de cartão de crédito é uma despesa. Ao cadastrar uma despesa com uma fatura, Os campos: QtdeParcelas e ValorTotal devem ser preenchidos como campos obrigatórios.  A aplicação deve criar despesas do tipo Cartão de crédito com o valor como parcelado, utilizando a fórmula:  ValorParcelado = ValorTotal / QdteParcelas  A aplicação deve criar as despesas nas próximas faturas correspondentes de acordo com a quantidade de parcelas.  Se não existir uma fatura, a aplicação deve criar a fatura de acordo com o mês e ano correspondente. |
| **[RN03] -** Cadastrar receitas no sistema | Ao cadastrar uma receita e defini-la como recebida, a aplicação deve acrescentar o valor do saldo de contas com o valor da receita recebida, como na formula abaixo:  saldoConta = saldoConta + ValorReceita |
| **[RN04] -** Efetua pagamento de despesa | Ao cadastrar uma despesa e defini-la como paga, a aplicação deve acrescentar o valor do saldo de contas com o valor da receita recebida, como na formula abaixo:  saldoConta = saldoConta - ValorDespesa |
| **[RN05] -** Efetua recebimento de receita | Ao definir uma receita como recebida, a aplicação deve acrescentar o valor do saldo de contas com o valor da receita recebida, como na formula abaixo:  saldoConta = saldoConta + ValorReceita |
| **[RN06] –** Consulta saldos | O sistema ira listar todos os saldos de contas, total de receitas e despesas do mês atual. |

# Requisitos

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisitos** | **Tipo (Funcional, não funcional)** | **Importância** | **Complexidade** |
| **[RF001] –** A aplicação deve prover ao usuário a opção de cadastrar novas despesas | Funcional | ALTA | MEDIA |

|  |  |  |  |
| --- | --- | --- | --- |
| **[RF002] -** A aplicação deve prover ao usuário a opção de cadastrar novas receitas | Funcional | ALTA | ALTA |
| **[RF003] -** A aplicação deve prover ao usuário a opção de registrar pagamentos de despesas | Funcional | MEDIA | MEDIA |
| **[RF004] -** A aplicação deve prover ao usuário a opção de registrar recebimentos de receitas | Funcional | ALTA | MEDIA |
| **[RF005] -** A aplicação deve prover ao usuário a listagem de despesas do mês | Funcional | ALTA | BAIXA |
| **[RF006] -** O sistema deve permitir que o usuário insira quantas despesas e receitas achar necessário | Funcional | ALTA | BAIXA |
| **[RNF007] -** O sistema web deve ser desenvolvido na linguagem c#, utilizando plataforma .Net | Não Funcional | MEDIA | MEDIA |
| **[RNF008] -** O sistema deve prover fácil  usabilidade | Não Funcional | ALTA | MEDIA |
| **[RNF009] -** O sistema só pode ser acessado com o usuário  previamente cadastrado com nome  de usuário e senha | Não Funcional | ALTA | BAIXA |
| **[RNF010] -** O sistema deve ter conexão com banco de dados SQL Server | Não Funcional | MEDIA | MEDIA |
| **[RNF011] -** O sistema deve utilizar o React Native para o desenvolvimento do aplicativo Mobile | Não Funcional | MEDIA | MEDIA |

# Matriz RF x UC

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **REQUISITOS\**  **CASOS DE USO X** | **UC001** | **UC002** | **UC003** | **UC004** | **UC005** | **UC006** |
| **[RF001] -** A aplicação deve prover ao usuário a opção de cadastrar novas despesas |  | **x** |  |  |  |  |
| **[RF002] -** A aplicação deve prover ao usuário a opção de cadastrar novas receitas |  |  | **x** |  |  |  |
| **[RF003] -** A aplicação deve prover ao usuário a opção de cadastrar pagamentos de despesas |  |  |  | **x** |  |  |
| **[RF004] -** A aplicação deve prover ao usuário a opção de cadastrar recebimentos de receitas |  |  |  |  | **x** |  |
| **[RF005] -** A aplicação deve prover ao usuário a listagem de despesas do mês |  | **x** |  |  |  | **x** |
| **[RF006] -** O sistema deve permitir que o usuário insira quantas despesas e receitas achar necessário |  | **x** | **x** |  |  |  |

# Matriz UC x RN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **REQUISITOS\**  **CASOS DE USO** | **UC001** | **UC002** | **UC003** | **UC004** | **UC005** | **UC006** |
| **[RNF007] -** O sistema web deve ser desenvolvido na linguagem c#, utilizando plataforma .Net | **X** | **X** | **X** | **X** | **X** | **X** |
| **[RNF008] -** O sistema deve prover fácil  usabilidade | **X** | **X** | **X** | **X** | **X** | **X** |
| **[RNF009] -** O sistema só pode ser acessado com o usuário  previamente cadastrado com nome  de usuário e senha | **X** |  |  |  |  |  |
| **[RNF010] -** O sistema deve ter conexão com banco de dados SQL Server | **X** | **X** | **X** | **X** | **X** | **X** |
| **[RNF011] -** O sistema deve utilizar o React Native para o desenvolvimento do aplicativo Mobile | **X** | **X** | **X** | **X** | **X** | **X** |