

Laboratório de Engenharia de Software

LES

Desenvolvimento de Software

Versão: 21/11/2017

*Prof. Antonio Sergio Bernardo*

RESUMO x

1. INTRODUÇÃO.......,..... x

2. ANÁLISE DE NEGÓCIOS

2.1 Análise da Situação Atual x

2.1.1 Descrição do Macro-Processo x

2.1.2 Diagrama de Processos As Is x

2.1.3 Diagnóstico da Situação Atual x

2.1.4 Descrição das Mudanças Necessárias x

2.2 Planejamento de Melhorias x

2.2.1 Diagrama de Processos To Be x

2.2.2 Descrição dos Processos To Be x

Fornecimento de Refeições x

Preparação de Refeições x

Recebimento de Mercadorias x

2.2.3 Plano de Implementação (Word) x

3. SISTEMA DE INFORMAÇÃO

3.1 Requisitos Não Funcionais....................................................................................................................... x

3.1.1 Requisitos de Desempenho............................................................................................................ x

3.1.2 Requisitos de Armazenamento x

3.1.3 Requisitos de Hwm Sw, Rede x

3.1.4 Requisitos Outros (Anexo) x

3.2 Requisitos Funcionais x

3.2.1 Diagramas de Casos de Uso x

3.2.2 Descrição dos Casos de Uso x

3.3 Projeto Detalhado do Software x

3.3.1 Arquitetura da Aplicação x

3.3.2 Tecnologias a Serem Utilizadas x

3.3.3 Componentes do Sw x

3.3.4 Modelo de Dados x

3.3.5 Protótipo Back-End x

3.3.6 Protótipo Front-End x

3.3.7 Refinamento dos Requisitos de Hw, Sw, Rede x

3.4 Codificação x

3.4.1 Script de Tabelas x

3.4.2 Código Fonte (CD Anexo) x

3.4.3 Testes e Aprovação x

4. GERENCIAMENTO DA TRANSIÇÃO

4.1 Preparação.............................................................................................................................................. x

4.1.1 Adequação de Recursos (Hw, Sw, Rede) x

4.1.2 Instalação do Software x

4.1.3 Configurações de Acesso e Permissões x

4.1.4 Aspectos Operacionais x

4.2 Efetivação ............................................................................................................................................ x

4.2.1 Treinamentos x

4.2.2 Implementação das Mudanças x

4.3 Finalização........................................................................................................................................... x

4.3.1 Monitoramento e Refinamento (Ajustes) x

4.3.2 Planejamento de Novo Ciclo (Versão) x

5. CONCLUSÃO x

6. BIBLIOGRAFIA x

7. GLOSSÁRIO x

**Nome do Projeto**

RESUMO

Este texto é um exemplo, use como modelo para a formação do seu TCC, não leve em consideração o conteúdo do texto aqui apresentado, trata-se apenas de um exemplo. Use para o Resumo letra Arial, tamanho 12, com parágrafo de espaçamento 1,5. Exatamente como este modelo.

Tente descrever no resumo o problema a ser abordado, as técnicas e ferramentas utilizadas no estudo e uma breve descrição do estudo de caso. O resumo não deve ser muito longo, tente escrever no máximo 4 parágrafos, o objetivo do resumo é mostrar para o leitor qual é o escopo do TCC.

Esta página deverá conter apenas o resumo.

1. INTRODUÇÃO

No conceito de BPM - Business Process Management, desenvolvimento de sistemas não se limita a Sistemas de Informação, que são apenas Recursos Opcionais dos Processos de Negócio e das Tarefas que os constituem

O Desenvolvimento, em sua integridade, consiste de uma Melhoria feita "Seletivamente" e "Sucessivamente" nos processos de negócio.

"Seletivamente", diagnosticando o ponto fraco da corrente, a partir da Cadeia de Valor, localizando o nível necessário para executar a transformação: Macro-Processo, Processo ou Tarefa, constituindo a abordagem estratégica Por Processos.

"Sucessivamente", porque geralmente não é viável, ou até mesmo não é possível fazer todas as melhorias desejáveis  de uma só vez. Assim temos a abordagem de Melhorias Contínuas do Ciclo PDCA, constituindo a abordagem De Processos.

Focamos neste trabalho o desenvolvimento de um sistema, trabalhado no contexto da Transformação de um Macro-Processo, envolvendo seus Processos Detalhados (conjuntos de tarefas) e seus Recursos (Funcionalidades do Sistema de Informação).

1. ANÁLISE DE NEGÓCIOS
   1. Análise da Situação Atual

2.1.1 Descrição do Macro-Processo Cantina

. Requisito Tipo: Map (Macro-Processo) As Is

. Sub Requisitos As Is

- Fornecimento de Refeições (Processo Primário)

- Preparação de Refeições (Processo Primário)

Nome: Cantina Brasil

Telefone: (15) 3333-4444

Site: Não tem

E-mail: cantinabrasil@hotmail.com

Horário de funcionamento: todos os dias das 8h às 23h

Formas de pagamento: Dinheiro, Cartões de Crédito e Vale-Refeições.

Não aceita cheques

Perfil: A Cantina Brasil serve comida caseira. As porções são os destaques

no jantar, peça calabresa com cebola, calabresa com batata frita e frango a

passarinho. Para beber oferece refrigerantes e sucos, entre outras opções.

Capacidade: Espaço para 30 pessoas. Área externa para 50 pessoas

Estacionamento: Não tem

Crianças: Não tem menu infantil. Não tem área infantil

Área para fumantes: É proibido o uso de cigarro, cachimbo ou cigarro de

palha

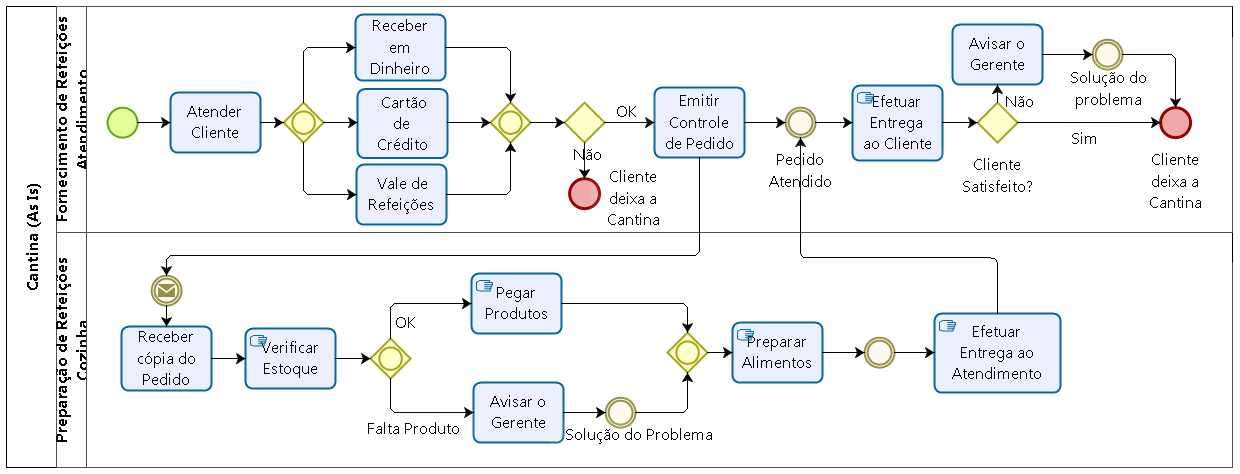
Não tem área para fumantes

Acesso para deficientes: Possui rampa de acesso sem auxílio de

corrimão.

Não tem banheiro adaptado

* + 1. Diagrama de Processos As Is



* + 1. Diagnóstico da Situação Atual

Nome da Análise: Revisão de Procedimentos

Descrição:

No Atendimento:

1 - A máquina de cartão de crédito utiliza linha telefônica compartilhada. Muitas vezes o cliente precisa esperar até que o gerente da cantina termine um telefonema. E às vezes desiste de comprar.

2 - Quando o Cliente não fica satisfeito e reclama do atendimento, a solução é ad-hoc (avisar o gerente). O gerente é sempre muito prestativo, mas quando não está presente, ninguém sabe o que fazer. E assim perdem muitos clientes.

Na Cozinha:

3 - A cópia do Controle de pedido muitas vezes se estravia. O cliente reclama da demora, e eventualmente alguem encontra o papel. Mas já se produziu mais um estrago.

4 - Se falta de um item no Estoque, o procedimento é também ad-hoc (avisar o gerente). Estando ele presente, imediatamente providencia o produto (corre ao supermercado vizinho). Mas estando ausente, o transtorno é grande.

* 1. Planejamento de Melhorias
     1. Considerações Legais
     2. Políticas organizacionais
     3. Especificação das Mudanças

Nome da Análise: Revisão de Procedimentos

Descrição:

1 -Instalar linha telefônica exclusiva para cartões de crédito.

2 - Examinar os problemas recorrentes que dependem do gerente, e eliminar

as causas.

As Melhorias estão especificadas no Modelo Proposto (To Be)

a) Controle de Pedidos será informatizado, evitando os frequentes extravios

de pedidos em papel.

b) Clientes insatisfeitos terão tratamento nas novas atividades:

- Resolver o Problema - Funcionários treinados conhecerão alternativas

para cada tipo de problema que venha a ocorrer.

- Registrar a Ocorrência - Estes registros orientarão futuras melhorias no

Processo.

c) Parcerias com Fornecedores selecionados para suprimento seguro dos

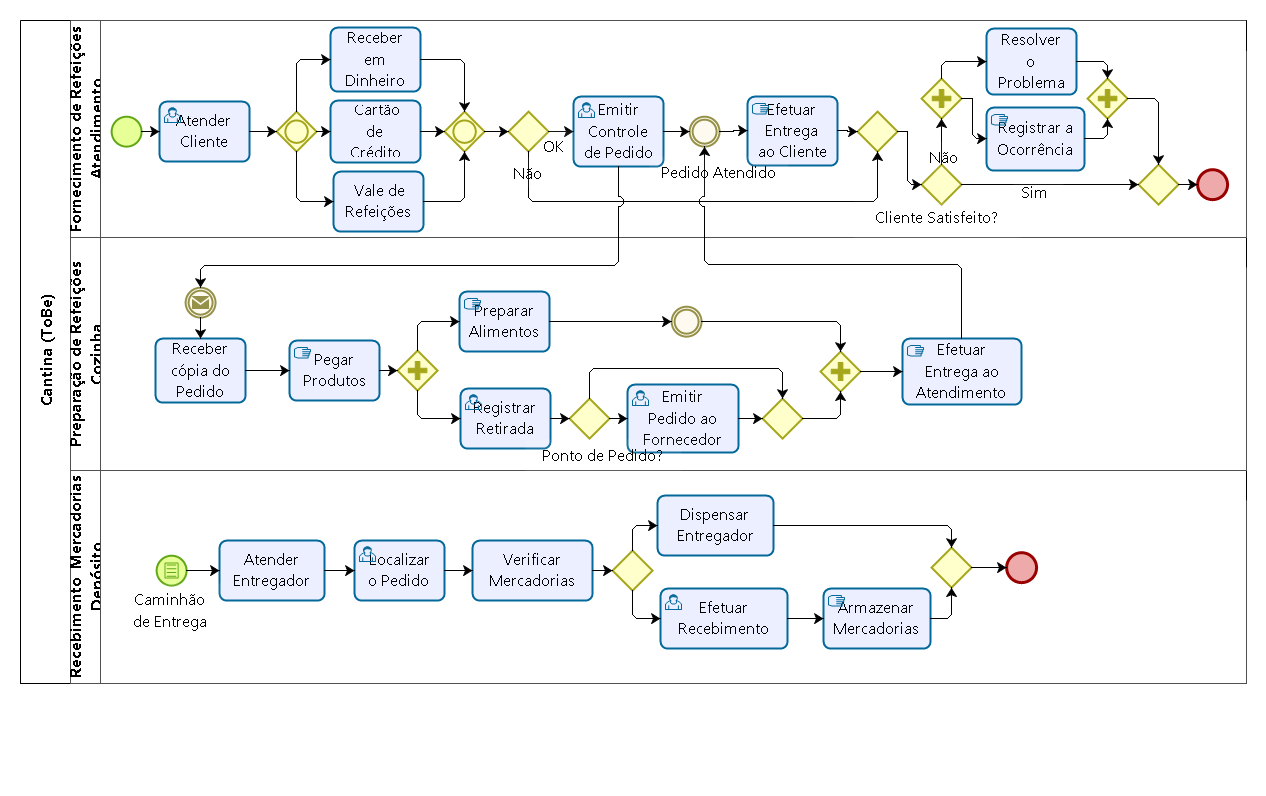
Itens.

- Haverá um Sistema Simples de Controle de Estoques, com alertas de Estoque Mínimo, evitando a falta completa de itens e os consequentes improvisos ou perdas de vendas.

3 - Elaborar e disponibilizar um Manual de Procedimentos para evitar a

excessiva dependência do gerente.

2.2.4 Diagrama de Processos To Be



2.2.5 Descrição dos Processos To Be

Fornecimento de Refeições

Preparação de Refeições

Recebimento de Mercadorias

* + 1. Plano de Implementação

1 A linha telefônica exclusiva para cartões de crédito evitará gargalos no Caixa, e o fluxo certamente vai melhorar, evitando interrupções no processo. No entanto a preparação dos lanches terá o volume aumentado. Portanto a nova linha telefônica deverá ser instalada junto com as outras melhorias.

2 Com a eliminação das causas recorrentes dos problemas antigos, e com o Processo To Be tornando explícitas e visíveis as Atividades, o Manual de Procedimentos irá complementar as melhorias dando mais segurança a todos os funcionários, e liberando o gerente para gerenciar a cantina mais estrategicamente, incluindo sua presença atenciosa e simpática para com os clientes, aproveitando para observar e obter feed-backs.

3 Na primeira semana do novo processo, o gerente deverá estar presente e atento em relação aos funcionários, não mais para soluções ad-hoc, e sim para habilitar e orientar as Atividades, tanto na rotina comum como nos cenários de exceção que já estarão previstos e ensaiados, e tendo ainda o reforço do Manual de Procedimentos.

4 A partir da segunda semana, o gerente deverá observar os funcionários mais à distância, para avaliar as mudanças efetuadas, e orientar sem tirar as responsabilidades.

5 Na consolidação do Novo Processo, o gerente deve estimular a Melhoria Contínua. Os funcionários estarão trabalhando com mais consciência dos Processos, e de como podem contribuir para melhora-los sempre mais.

6 O Consultor de BPM que foi contratado para as mudanças efetuadas, terá um contrato para visitas e observações periódicas, pois há indicações de que os negócios podem ser expandidos.

1. SISTEMA DE INFORMAÇÃO

3.1 Requisitos Não Funcionais

3.1.1 Requisitos Organizacionais

RNF1 – Compatibilidade com navegadores Internet

O sistema funcionará nos navegadores de internet Internet Explorer, Google Chrome e Mozilla Firefox.

3.1.2 Requisitos de Produto

RNF2 - Carga rápida de dados

Carregamento rápido de dados do servidor, as páginas do sistema devem carregar no máximo em 3 segundos.

RNF3 - Espaço

A aplicação rodará via web, e possui informações que são armazenadas em um datacenter, portanto não é necessário um hardware específico para que o sistema seja operado, bastando ter apenas conexão com internet, para pode operar o sistema.

RNF4 – Interface Intuitiva

Interface desenvolvida para facilitar a agilidade e seu uso.

3.1.3 Requisitos Externos

RNF5 – Acesso e alteração do banco de dado somente por usuários com permissão

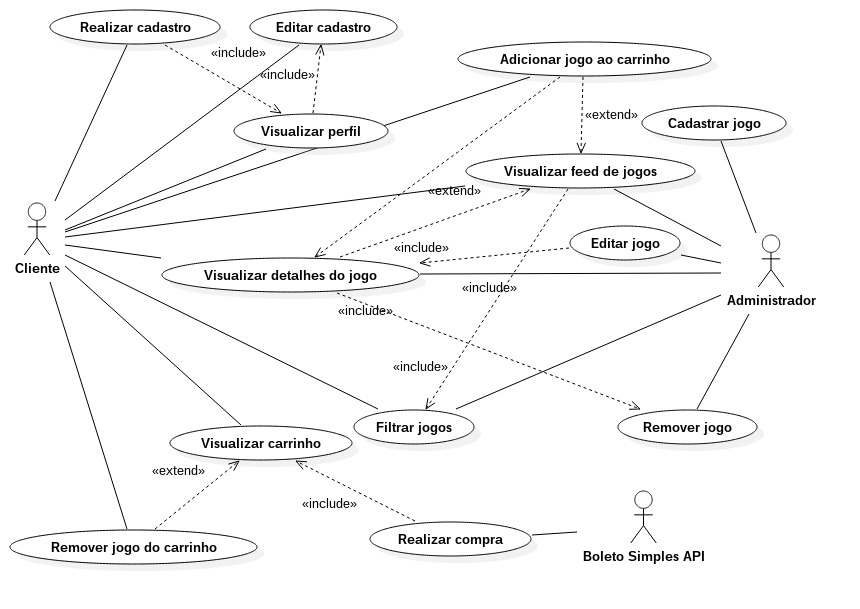
Todo e qualquer dado do banco de dados, só poderá ser alterado por aquela com permissão para tal.

RNF6 – Privacidade

Todo e qualquer dado inserido no banco de dado, que, referente aos dados do usuário, por razões legais, não serão divulgados.

* 1. Requisitos Funcionais

3.2.1 Diagramas de Casos de Uso



3.2.2 Descrição dos Casos de Uso

|  |
| --- |
| **Caso de uso:** Realizar cadastro |
| **Atores:** Cliente |
| **Interessados:** Cliente que deseja ter acesso as compras |
| **Pré-condições:** Cliente não possui cadastro no sistema |
| **Pós-condições:** Usuário é redirecionado para o feed de jogos |
| **Requisitos correlacionados:** |
| **Variações tecnológicas:** |
| **Questões em aberto** |
| **Fluxo principal:**   1. O cliente entra em cadastro de usuário 2. O cliente insere seus dados pessoais 3. O sistema valida os dados 4. O cliente salva os dados no sistema |
| **Tratamento de exceções:**  **2.** **Os dados estão inválidos**  2.1 Volta passo 2 |

|  |
| --- |
| **Caso de uso:** Editar cadastro |
| **Atores:** Cliente |
| **Interessados:** Cliente que deseja atualizar seu cadastro |
| **Pré-condições:** Cliente deve possuir cadastro no sistema |
| **Pós-condições:** Mensagem de atualização é mostrada ao usuário |
| **Requisitos correlacionados:** |
| **Variações tecnológicas:** |
| **Questões em aberto** |
| **Fluxo principal:**   1. O cliente visualiza o perfil 2. O cliente insere seus dados pessoais 3. O sistema valida os dados 4. O cliente salva os dados no sistema |
| **Tratamento de exceções:**  **2.** **Os dados estão inválidos**  2.1 Volta passo 2 |

|  |
| --- |
| **Caso de uso:** Visualizar perfil |
| **Atores:** Cliente |
| **Interessados:** Cliente que deseja visualizar seu perfil |
| **Pré-condições:** Estar cadastrado no sistema |
| **Pós-condições:** Perfil é mostrado ao usuário |
| **Requisitos correlacionados:** |
| **Variações tecnológicas:** |
| **Questões em aberto** |
| **Fluxo principal:**   1. O cliente entra em visualizar perfil |
| **Tratamento de exceções:** |

|  |
| --- |
| **Caso de uso:** Adicionar jogo ao carrinho |
| **Atores:** Cliente |
| **Interessados:** Cliente que deseja comprar um jogo |
| **Pré-condições:** O cliente deve possuir cadastro no sistema |
| **Pós-condições:** O cliente deve poder continuar a navegação |
| **Requisitos correlacionados:** |
| **Variações tecnológicas:** |
| **Questões em aberto** |
| **Fluxo principal:**   1. O cliente seleciona adicionar ao carrinho 2. O jogo é adicionado ao carrinho |
| **Tratamento de exceções:** |

|  |
| --- |
| **Caso de uso:** Cadastrar jogo |
| **Atores:** Administrador |
| **Interessados:** O administrador do sistema |
| **Pré-condições:** O administrador deve possuir todos os dados do jogo |
| **Pós-condições:** Mensagem de jogo cadastrado deve ser mostrado |
| **Requisitos correlacionados:** |
| **Variações tecnológicas:** |
| **Questões em aberto** |
| **Fluxo principal:**   1. O administrador deve selecionar cadastrar jogo 2. O administrador deve inserir os dados 3. O sistema deve validar os dados 4. O administrador salva os dados |
| **Tratamento de exceções**  **2 O sistema deve validar os dados**  2.1 Volta ao passo 2 |

* 1. Projeto Detalhado do Sofware

3.3.1 Arquitetura da Aplicação

3.3.2 Tecnologias a Serem Utilizadas

As tecnologias usadas nesse projeto são: Angular, MongoDB e NODE.js

3.3.3 Componentes do Sw

#### /users

* **POST**

Inserts a user into database. A Json body is required as shown next.

$ curl -s -XPOST "localhost:8080/users" -H "Content-Type: application/json" -d '{

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1

}' | jq

{

"\_id": "5afba392c390fa4df05ec85b",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

}

#### /users/:userId

* **PUT**

Change a user's data. Id must be passed as param (**:userId**), a Json body with location data is also required as shown.

$ curl -s -XPUT "localhost:8080/users/5afba392c390fa4df05ec85b" -H "Content-Type: application/json" -d '{

"email": "example@changed.com",

"username": "anotherexample",

"password": "examplero",

"type": 2

}' | jq

{

"\_id": "5afba392c390fa4df05ec85b",

"email": "example@changed.com",

"username": "anotherexample",

"password": "examplero",

"type": 2,

"\_\_v": 0

}

#### /users

* **GET**

get all users data

$ curl -s "localhost:8080/users" -H "Content-Type: application/json" | jq

[

{

"\_id": "5b2194cbe383a5303d0c8d0b",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

},

{

"\_id": "5b2195b10ec977309985cb5f",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

}

]

#### /users/:userId

* **GET**

get single user data. userId's data.

$ curl -s "localhost:8080/users/5b2194cbe383a5303d0c8d0b" -H "Content-Type: application/json" | jq

{

"\_id": "5b2194cbe383a5303d0c8d0b",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

}

### /users/:userId

* **DELETE**

delete specified users data from db.

$ curl -s -XDELETE "localhost:8080/users/5b2194cbe383a5303d0c8d0b" -H "Content-Type: application/json" | jq

{

"message": "User deleted."

}

## Login

#### /login

* **POST**

send user's username and login. if it matches with db data, returns users data.

$ curl -s -XPOST "localhost:8080/login" -H "Content-Type: application/json" -d '{

"username": "example",

"password": "example123"

}' | jq

{

"\_id": "5b2195b10ec977309985cb5f",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

}

wrong input

{

"error": "username or password incorrect."

}

## Games endpoints

#### /games

* **POST**

Inserts a game into database. A Json body is required as shown next.

$ curl -s -XPOST "localhost:8080/games" -H "Content-Type: application/json" -d '{

"name": "corrida rapida",

"synopse": "corra o mais rapido ",

"tags": "Corrida",

"price": 12.35,

"releasedate": "2013-03-01T01:10:00",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "Corrida",

"rating": 0,

"imgsrc": "https://images.livrariasaraiva.com.br/imagemnet/imagem.aspx/?pro\_id=9171307&qld=90&l=430&a=-1"

}' | jq

{

"\_id": "5b22fd4960860437638be767",

"name": "corrida rapida",

"synopse": "corra o mais rapido ",

"tags": "Corrida",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "Corrida",

"rating": 0,

"imgsrc": "https://images.livrariasaraiva.com.br/imagemnet/imagem.aspx/?pro\_id=9171307&qld=90&l=430&a=-1",

"\_\_v": 0

}

#### /games/:gameId

* **PUT**

Change a games's data. Id must be passed as param (**:gameId**), a Json body with location data is also required as shown.

$ curl -s -XPUT "localhost:8080/games/5b22f72bee1b383613e6c8f2" -H "Content-Type: application/json" -d '{

"name": "sanic gotta go slow",

"synopse": "ouriço",

"tags": "corrida",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "horror",

"rating": 0,

"imgsrc": "https://http2.mlstatic.com/sonic-forces-ps4-midia-fisica-lacrado-novo-original-D\_NQ\_NP\_658382-MLB26464235963\_112017-F.webp"

}' | jq

{

"\_id": "5b22f72bee1b383613e6c8f2",

"name": "sanic gotta go slow",

"synopse": "ouriço",

"tags": "corrida",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "horror",

"rating": 0,

"imgsrc": "https://http2.mlstatic.com/sonic-forces-ps4-midia-fisica-lacrado-novo-original-D\_NQ\_NP\_658382-MLB26464235963\_112017-F.webp",

"\_\_v": 0

}

#### /games

* **GET**

get all games data

$ curl -s "localhost:8080/games" -H "Content-Type: application/json" | jq

[

{

"\_id": "5b22f72bee1b383613e6c8f2",

"name": "sanic gotta go slow",

"synopse": "ouriço",

"tags": "corrida",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "horror",

"rating": 0,

"imgsrc": "https://http2.mlstatic.com/sonic-forces-ps4-midia-fisica-lacrado-novo-original-D\_NQ\_NP\_658382-MLB26464235963\_112017-F.webp",

"\_\_v": 0

},

{

"\_id": "5b22f8c286969c3646ef8c41",

"name": "Roller Coaster",

"synopse": "Construa seu parque",

"tags": "Construcao",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "Construçao",

"rating": 0,

"imgsrc": "https://http2.mlstatic.com/rollercoaster-tycoon-world-dvd-pc-midia-fisica-D\_NQ\_NP\_782650-MLB26799855909\_022018-F.webp",

"\_\_v": 0

}

]

#### /games/:gameId

* **GET**

get single game data. gameId's data.

$ curl -s "localhost:8080/games/5b22f8c286969c3646ef8c41" -H "Content-Type: application/json" | jq

{

"\_id": "5b22f8c286969c3646ef8c41",

"name": "Roller Coaster",

"synopse": "Construa seu parque",

"tags": "Construcao",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "Construçao",

"rating": 0,

"imgsrc": "https://http2.mlstatic.com/rollercoaster-tycoon-world-dvd-pc-midia-fisica-D\_NQ\_NP\_782650-MLB26799855909\_022018-F.webp",

"\_\_v": 0

}

### /games/:gameId

* **DELETE**

delete specified games data from db.

$ curl -s -XDELETE "localhost:8080/games/5b22f8c286969c3646ef8c41" -H "Content-Type: application/json" | jq

{

"message": "Game deleted."

}

#### /genres

* **GET**

get all genres from games

$ curl -s "localhost:8080/genres" -H "Content-Type: application/json" | jq

[

"horror",

"Corrida"

]

3.3.4 Modelo de Dados

User:

"\_id": "5afba392c390fa4df05ec85b",

"email": "example@example.com",

"username": "example",

"password": "example123",

"type": 1,

"\_\_v": 0

Game:

"\_id": "5b22fd4960860437638be767",

"name": "corrida rapida",

"synopse": "corra o mais rapido ",

"tags": "Corrida",

"price": 12.35,

"releasedate": "2013-03-01T04:10:00.000Z",

"developer": "lero dvpl",

"editor": "lero edts",

"genre": "Corrida",

"rating": 0,

"imgsrc": "https://images.livrariasaraiva.com.br/imagemnet/imagem.aspx/?pro\_id=9171307&qld=90&l=430&a=-1",

"\_\_v": 0

3.3.5 Protótipo Front-End

3.3.6 Protótipo Back-End

3.3.6.1 Routes

3.3.6.1.1 GameRoutes.js

|  |
| --- |
| 'use strict'; |
|  |  |
|  | module.exports = function(app) { |
|  | var Game = require('../controllers/GameController'); |
|  |  |
|  | //game routes "for bunch of data" |
|  | app.route('/games') |
|  | .get(Game.list\_all\_games) |
|  | .post(Game.insert\_a\_game); |
|  |  |
|  | //game routes "for sigle data" |
|  | app.route('/games/:gameId') |
|  | .get(Game.read\_a\_game) |
|  | .put(Game.update\_a\_game) |
|  | .delete(Game.delete\_a\_game); |
|  |  |
|  | //game routes "for bunch of data" |
|  | app.route('/genres') |
|  | .get(Game.list\_all\_genres); |
|  | }; |

3.3.6.1.2 UserRoutes.js

|  |
| --- |
| 'use strict'; |
|  |  |
|  | module.exports = function(app) { |
|  | var User = require('../controllers/UserController'); |
|  |  |
|  | //user routes "for bunch of data" |
|  | app.route('/users') |
|  | .get(User.list\_all\_users) |
|  | .post(User.insert\_a\_user); |
|  |  |
|  | //user routes "for sigle data" |
|  | app.route('/users/:userId') |
|  | .get(User.read\_a\_user) |
|  | .put(User.update\_a\_user) |
|  | .delete(User.delete\_a\_user); |
|  |  |
|  | //login data |
|  | app.route('/login') |
|  | .post(User.login); |
|  | }; |

3.3.6.2 Models

3.3.6.2.1 GameModel.js

|  |
| --- |
| 'use strict'; |
|  |  |
|  | var mongoose = require('mongoose'); |
|  | var Schema = mongoose.Schema; |
|  |  |
|  | //creating game's base struct |
|  | var GameSchema = new Schema({ |
|  | name: { |
|  | type: String, |
|  | required: 'Enter the game\'s name' |
|  | }, |
|  | synopse: { |
|  | type: String |
|  | }, |
|  | tags: { |
|  | type: String |
|  | }, |
|  | price: { |
|  | type: Number |
|  | }, |
|  | releasedate: { |
|  | type: Date |
|  | }, |
|  | developer: { |
|  | type: String |
|  | }, |
|  | editor: { |
|  | type: String |
|  | }, |
|  | genre: { |
|  | type: String |
|  | }, |
|  | rating: { |
|  | type: Number |
|  | }, |
|  | imgsrc: { |
|  | type: String |
|  | } |
|  | }); |
|  |  |
|  | module.exports = mongoose.model('Games', GameSchema); |

3.3.6.2.2 UserModel.js

|  |
| --- |
| 'use strict'; |
|  |  |
|  | var mongoose = require('mongoose'); |
|  | var Schema = mongoose.Schema; |
|  |  |
|  | //creating user's base struct |
|  | var UserSchema = new Schema({ |
|  | email: { |
|  | type: String, |
|  | required: 'Enter the user\'s email' |
|  | }, |
|  | username: { |
|  | type: String |
|  | }, |
|  | password: { |
|  | type: String |
|  | }, |
|  | type: { |
|  | type: Number |
|  | } |
|  | }); |
|  |  |
|  | module.exports = mongoose.model('Users', UserSchema); |

3.3.6.3 Controllers

3.3.6.3.1 GameController.js

|  |
| --- |
| 'use strict'; |
|  |  |
|  | var mongoose = require('mongoose'); |
|  | Game = mongoose.model('Games'); |
|  |  |
|  | //GET at /games |
|  | exports.list\_all\_games = function(req, res) { |
|  | Game.find({}, function(err, game) { |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(game); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //POST at /games |
|  | exports.insert\_a\_game = function(req, res) { |
|  | var new\_game = new Game(req.body); |
|  | new\_game.save(function(err, game){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } |
|  | res.status(201).json(game); |
|  | }); |
|  | }; |
|  |  |
|  | //GET at /games/:gameId |
|  | exports.read\_a\_game = function(req, res) { |
|  | Game.findById(req.params.gameId, function(err, game) { |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(game); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //PUT at /games/:gameId |
|  | exports.update\_a\_game = function(req, res) { |
|  | Game.findOneAndUpdate( |
|  | {\_id: req.params.gameId}, req.body, {new: true}, function(err, game){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  |  |
|  | res.status(200).json(game); |
|  | } |
|  | }) |
|  | } |
|  |  |
|  | //DELETE at /games/:gameId |
|  | exports.delete\_a\_game = function(req, res) { |
|  | Game.remove({\_id: req.params.gameId}, function(err, game){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json({message: "Game deleted."}); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //GET at /genres |
|  | exports.list\_all\_genres = function(req, res) { |
|  | Game.distinct("genre", function(err, game) { |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(game); |
|  | } |
|  | }); |
|  | }; |

**3.3.6.3.2 UserController.js**

|  |
| --- |
| 'use strict'; |
|  |  |
|  | var mongoose = require('mongoose'); |
|  | User = mongoose.model('Users'); |
|  |  |
|  | //GET at /users |
|  | exports.list\_all\_users = function(req, res) { |
|  | console.log('alan') |
|  | User.find({}, function(err, user) { |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(user); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //POST at /users |
|  | exports.insert\_a\_user = function(req, res) { |
|  | var new\_user = new User(req.body); |
|  | new\_user.save(function(err, user){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(201).json({"id": new\_user.\_id, |
|  | "email": new\_user.email, |
|  | "username": new\_user.username, |
|  | "type": new\_user.type}); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //GET at /users/:userId |
|  | exports.read\_a\_user = function(req, res) { |
|  | User.findById(req.params.userId, function(err, user) { |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(user); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //PUT at /users/:userId |
|  | exports.update\_a\_user = function(req, res) { |
|  | User.findOneAndUpdate( |
|  | {\_id: req.params.userId}, req.body, {new: true}, function(err, user){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json(user); |
|  | } |
|  | }) |
|  | } |
|  |  |
|  | //DELETE at /users/:userId |
|  | exports.delete\_a\_user = function(req, res) { |
|  | User.remove({\_id: req.params.userId}, function(err, user){ |
|  | if (err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | res.status(200).json({message: "User deleted."}); |
|  | } |
|  | }); |
|  | }; |
|  |  |
|  | //Login |
|  | exports.login = function(req, res) { |
|  | User.findOne({username: req.body.username}, function(err, user){ |
|  | if(err){ |
|  | res.status(500).send(err); |
|  | } else { |
|  | if(user != null) |
|  | { |
|  | if(req.body.password == user.password) |
|  | { |
|  | res.status(200).json(user); |
|  | } else { |
|  | res.status(401).json({error: "username or password incorrect."}) |
|  | } |
|  | } else { |
|  | res.status(401).json({error: "username or password incorrect."}) |
|  | } |
|  |  |
|  | } |
|  | }) |
|  | } |