DBSLP - Database System Learning Platform

Versão 1.0.0

2023

MANUEL VINÍCIUS SOUZA BRANDÃO FIGUEIRÊDO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DBSLP - DATABASE SYSTEM LEARNING PLATFORM

Este documento foi apresentado como requisito para obtenção de nota pelo 5º semestre da graduação em ciência da computação pela Unijorge-Campus Paralela, sob orientação dos discentes Celso Barreto, Fábio Gomes, Sheila Tirony e Jose Vicente.

SALVADOR, BAHIA

2023

Sumário

1. Introdução 4

1.1 Finalidade 4

1.2 Escopo 4

2 Descrição geral do sistema 5

2.1 Sobre o Problema 5

2.3 Principais envolvidos e suas características 6

2.3.1 Usuários do sistema 6

2.3.2 Desenvolvedores do Sistema 6

3 Descrição geral da arquitetura de desenvolvimento 7

4 Cenários de Casos de Uso (Visão) 8

4.1 Realizações de casos de uso 9

4.1.1 Carregar estado das modelagens 9

4.1.2 Carregar um Banco de Dados 10

4.1.3 Conectar em uma Sala 11

4.1.4 Desconectar de uma Sala 11

4.1.5 Executar SQL Scripts 11

4.1.6 Executar um SGBD 12

4.1.7 Modelar Sistemas de Banco de Dados 12

4.1.8 Salvar as modelagens como PNG 13

4.1.9 Salvar o Banco de Dados 13

4.1.10 Salvar o estado das modelagens 14

4.1.11 Selecionar uma sala 14

4.1.12 Terminar um SGBD 15

4.1.13 Verificar a presença de anomalias nas Tabelas 16

4.2 Protótipos 16

5 Visão Lógica 22

5.1 Aplicação de modelagem 22
5.2 Aplicação do laboratório SQL 33
6 Visão de Processos 34
7 Visão de Implementação 35
8 Visão de Implantação 36
9 Licenças 37
10 Testes 41
11 Problemas conhecidos 42
REFERÊNCIAS 43
ANEXOS 44
ANEXO I – Diagrama de Classes 45

1. Introdução

1.1 Finalidade

Este documento apresenta os requerimentos do sistema e quais decisões técnicas foram tomadas para atendê-las, servindo como guia de implementação para os engenheiros e desenvolvedores.

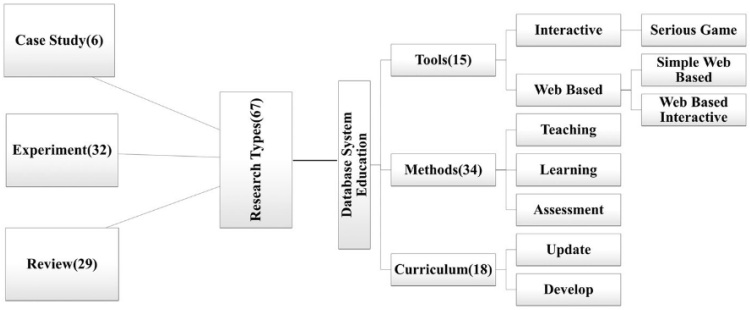
1.2 Escopo

Este documento abrange: (i) a definição da missão ou objetivo geral do projeto e suas expectativas (ii) a definição dos requerimentos ou funções (iii) protótipos de alta fidelidade de sua interface (iv) quais decisões técnicas foram tomadas para implementá-los, isso é, a sua arquitetura (v) quais as limitações conhecidas (vi) quais as expectativas para o futuro do projeto.

2 Descrição geral do sistema[1]

2.1 Sobre o Problema

As soluções para problemas de ensino-aprendizado de banco de dados geralmente são de três tipos: (i) ferramentas (ii) métodos (iii) currículos, conforme mostra a compilação de Muhammad *et al*. (2023) na Figura 1.



Fonte: Muhammad et al (2023, p.13).

Figura 1 Taxonomia dos estudos sobre Database System Education.

O uso de ferramentas é feito por diversas abordagens, como modelagem, verificação de scripts, etc. Contudo,

Assim, quando essas são especializadas e autossuficientes, esses sistemas se tornam de difícil integração, o que afeta a experiência dos usuários e a experiência de desenvolvedores que buscam integrá-las (BRUSILOVSKY *et al*., 2010).

Entre os problemas comuns para ensino-aprendizado de banco de dados, pode-se convenientemente ressaltar os seguintes, conforme Hamzah (2019) e Poletto, Santos e Júnior (2013): (i) que há dificuldade no design de banco de dados, especialmente normalização, mas também diagramas Entidade-Relacionamento (ii) que há pouca qualidade nos SQL dos discentes (iii) que entrevistas de empregos sugerem a falta de habilidade (*expertise*) de graduandos em banco de dados.

Dessa forma, um sistema que possibilita a prática de modelagem e de scripts SQL é fundamentado.

---

[1] Esta seção é não-normativa, ou seja, apenas informacional quanto à estrutura do sistema.

2.2 Missão

O DBSLP (*Database System Learning Platform*) é uma plataforma web distribuída que tem como objetivo auxiliar o aprendizado de sistemas de banco de dados relacionais. Esta plataforma provê um sistema que integra modelagem e execução banco de dados relacionais nativamente no *browser*. Além disso, possibilita a criação de sessões colaborativas *real-time*, possibilitando que discentes se conectem tanto com outros discentes quanto com os docentes, o que melhora a experiência de aprendizado.

Em suma, DBSLP possibilita que o seguinte seja feito diretamente em browser

- Modelagem de bancos de dados relacionais;

- Execução de um banco de dados;

- Criação e uso de sessões colaborativas;

2.3 Principais envolvidos e suas características

2.3.1 Usuários do sistema

O sistema será usado por discentes, docentes ou interessados em sistemas de banco de dados relacionais, que se supõe que sejam acostumados com tecnologia e computadores.

O acesso ao sistema é feito preferencialmente por computadores, mas também por dispositivos móveis.

2.3.2 Desenvolvedores do Sistema

As áreas de atuação no projeto podem ser divididas em:

1) Arquitetos, responsáveis pela definição dos requisitos do sistema e a elaboração da sua arquitetura/estrutura;

2) Desenvolvedores front-end, responsáveis pela implementação da aplicação do cliente;

3) Desenvolvedores back-ends, responsáveis pela implementação dos servidores que distribuem artefatos ou informações.

3 Descrição geral da arquitetura de desenvolvimento

O modelo de arquitetura usado foi o *View Model 4 + 1*, que apresenta o sistema por meio de views (visualizações), em que cada view representa o mesmo sistema com detalhes ou ênfases diferentes (HU, 2023).

Conforme Hu (2023), o sistema pode ser apresentado por meio de quatro + 1 visualizações diferentes que são construídas sobre os casos de usos, consistindo de:

(i) visualização lógica (*logic view*), que especifica a modelagem de objetos em sistemas orientados a objetos;

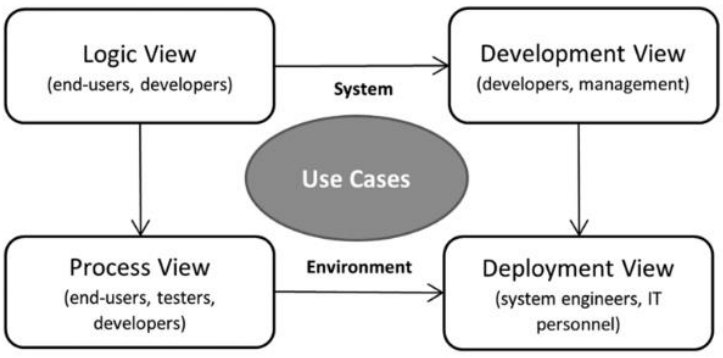
(ii) visualização de processo (*process view*), que especifica como as entidades concorrem e/ou cooperam pelos recursos seja de forma síncrona ou assíncrona.

(iii) visualização de desenvolvimento (*development view*), que especifica a organização estática do software durante o desenvolvimento, focando nos elementos de software de alto nível;

(iv) visualização de implantação (*deployment view*), que especifica toda a infraestrutura dos hardwares do sistema que executam os softwares e serviços implementado, ou seja, a topologia;

(v) cenários de caso de uso (+1 view), que especifica os requerimentos funcionais e provê as informações fundamentais para todas as outras visualizações;

A Figura 2 representa essa arquitetura.



Fonte: Hu (2023, p.289).

Figura 2 Representação de uma arquitetura *4 + 1 View Model*.

4 Cenários de Casos de Uso (Visão)

Os requisitos funcionais são:

- RF1: Prover ferramentas para modelagem conceitual e lógica;
- RF2: Prover mecanismo para sessões compartilhadas para a ferramenta de modelagem;
- RF3: As ferramentas de modelagens devem ser salváveis;
- RF4: As ferramentas de modelagens devem possibilitar exportação como Imagem;
- RF3: Prover um SGBD para execução de script SQL;

Os requisitos não-funcionais são:

- RNF1: A aplicação deve ser multiplataforma (web e mobile);
- RNF2: Deve ser acessível ao usuário a documentação do sistema;
- RNF3: A aplicação deve ser SPA (Single Page Application);

No Diagrama 1 está representado os casos de uso do sistema. Esse diagrama representa os requisitos funcionais como funcionalidades ou usos do sistema, sendo a sua realização dividida principalmente em três grandes pacotes: (i) lab, que consiste do ambiente para execução de SQL Scripts (ii) modeling, que consiste do ambiente para modelagem conceitual e lógica e (iii) hooks, que consistem dos serviços secundários ou complementares.

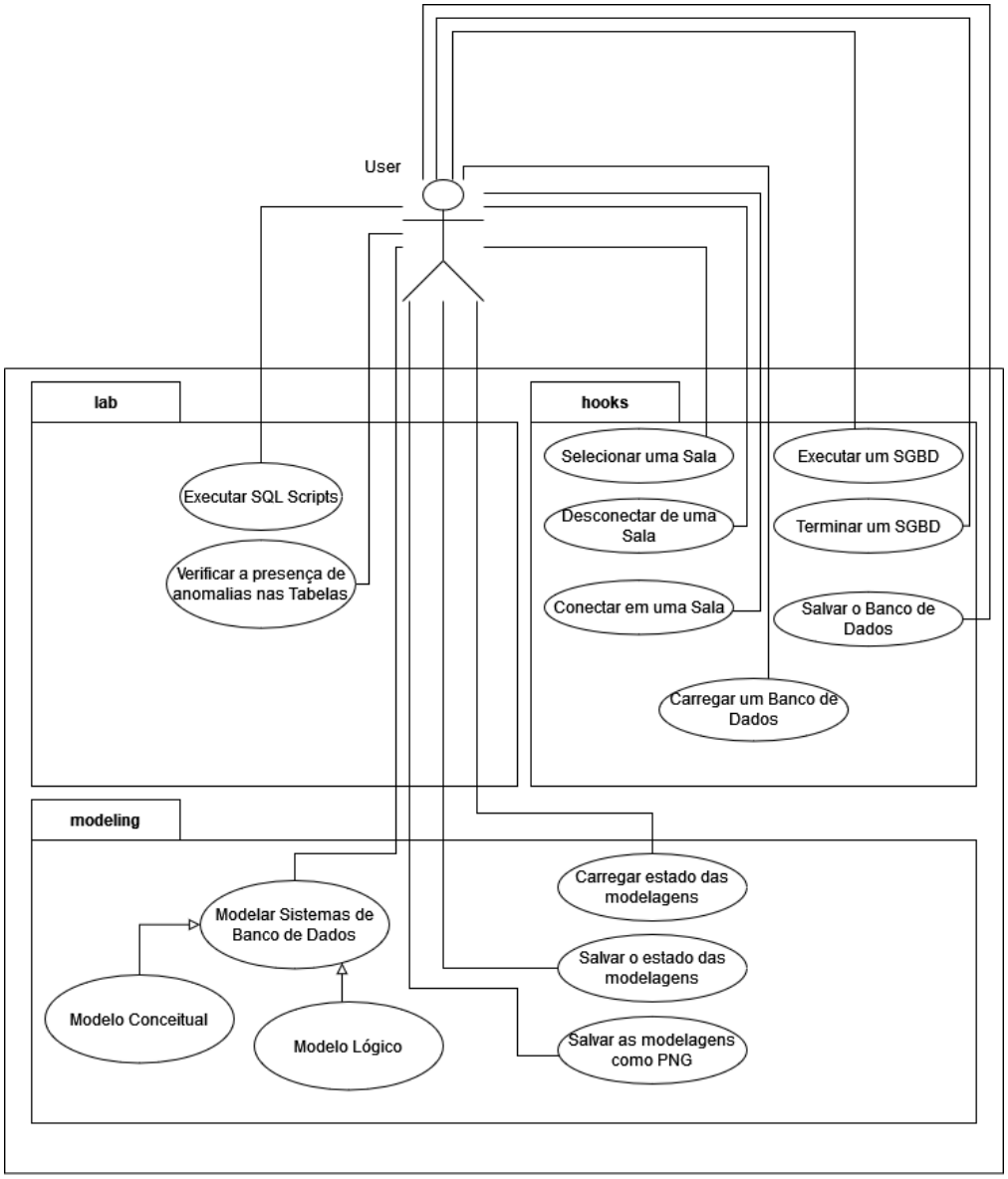


Diagrama 1 Casos de Uso.

4.1 Realizações de casos de uso

4.1.1 Carregar estado das modelagens

1 Objetivos

Permitir ao usuário carregar um documento de modelagem salvo previamente.

2 Sumário

O usuário pode carregar um documento previamente salvo (serializado), de forma que a sua sessão é restaurada.

3 Atores

Usuário

4 Fluxo Normal

a. Usuário clica no menu “Arquivos”;
b. Usuário clica no botão “Carregar Modelagem”;
c. O modal “Carregar Arquivos” é exibido;
d. Usuário clica no botão “Selecionar Arquivos”;
e. Usuário seleciona o arquivo JSON de seu computador;
f. Usuário clica no botão “Carregar”;
g. O arquivo é desserializado e é usado como novo estado da modelagem.

5 Fluxo Alternativo

a. Um formato de arquivo inválido é selecionado.

4.1.2 Carregar um Banco de Dados

1 Objetivos

Permitir ao usuário carregar um banco de dados salvo previamente.

2 Sumário

O usuário pode carregar um banco de dados previamente salvo (serializado), de forma que a sua sessão é restaurada.

3 Atores

Usuário.

4 Fluxo Normal

a. Usuário clica no botão “Sistema Gerenciador de Banco de Dados”;
b. O modal “Sistema Gerenciador de Banco de Dados” é exibido;
c. Usuário clica no botão “Selecionar Arquivos”
d. Usuário seleciona o arquivo BINARY de seu computador
e. Usuário clica no botão “Carregar”
f. O arquivo é desserializado e é usado como novo estado do banco de dados.

5 Fluxo Alternativo

b. Um formato de arquivo inválido é selecionado.

4.1.3 Conectar em uma Sala

1 Objetivos

Possibilitar ao usuário se conectar em uma sala de sessão compartilhada.

2 Sumário

O usuário pode entrar em uma sessão compartilhada a partir de seu token.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Gerenciar sessões”;
b. Usuário digita o token da sessão alvo;
c. Usuário clica em “Entrar na Sala”.

4.1.4 Desconectar de uma Sala

1 Objetivos

Possibilitar que o usuário se desconecte de uma sala de sessão compartilhada.

2 Sumário

O usuário pode se desconectar de uma sala de sessão compartilhada, não sincronizando o seu documento com o de outros participantes.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Gerenciar sessões”;
b. Usuário clica no botão “Desconectar”.

4.1.5 Executar SQL Scripts

1 Objetivos

Permitir ao usuário executar SQL Scripts.

2 Sumário

O usuário pode inserir SQL Scripts que serão executados por um SGBD, gerando um resultado, possivelmente.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário digita SQL Statements em um campo de texto;
b. Usuário clica no botão “Executar Script”;
c. Usuário recebe um feedback do sucesso ou não da execução;
d. Usuário recebe o resultado do Script, se algum.

5 Fluxo Alternativo

a. Ocorre erro na execução do script, não sendo mostrado algum resultado.

4.1.6 Executar um SGBD

1 Objetivos

Permitir ao usuário a execução de um processo de SGBD.

2 Sumário

O usuário pode começar a execução de um processo de SGBD, que é necessário para execução de SQL Scripts.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Sistema Gerenciador de Banco de Dados”;
b. O modal “Sistema Gerenciador de Banco de Dados” é exibido;
c. Usuário clica no botão “Executar”;
d. Um novo SGBD é alocado e é executado.

4.1.7 Modelar Sistemas de Banco de Dados

1 Objetivos

Permitir ao usuário ferramentas para modelagem conceitual e lógica de banco de dados relacionais.

2 Sumário

O usuário pode modelar conceitualmente e logicamente conforme os modelos Entidade-Relacionamento e relacional, respectivamente.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Modelagens”;
b. O aplicativo de modelagem é exibido.

4.1.8 Salvar as modelagens como PNG

1 Objetivos

Permitir ao usuário que salve uma modelagem específica (conceitual ou lógica) como PNG.

2 Sumário

O usuário pode salvar uma modelagem específica (conceitual ou lógica) como PNG.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no Menu “Arquivo”;
b. Usuário clica no botão “Salvar Como”;
c. Usuário clica no botão “PNG”.
d. O modelo atual (conceitual ou lógico) é salvo como PNG.

4.1.9 Salvar o Banco de Dados

1 Objetivos

Permitir ao usuário que salve (serialize) o estado atual do banco de dados.

2 Sumário

O usuário pode salvar (serializar) o estado atual do banco de dados, de forma que possa regenerá-lo (desserializar) posteriormente.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Sistema Gerenciador de Banco de Dados”;
b. O modal “Sistema Gerenciador de Banco de Dados” é exibido;
c. Usuário clica no botão “Salvar”
d. O estado do banco de dados atual é salvo como BINARY.

4.1.10 Salvar o estado das modelagens

1 Objetivos

Permitir ao usuário que salve (serialize) o estado atual das modelagens (conceitual e lógica).

2 Sumário

O usuário pode salvar (serializar) o estado atual das modelagens (conceitual e lógica), de forma que possa regenerá-lo (desserializar) posteriormente.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no menu “Arquivo”.
b. Um menu é exibido.
c. Usuário clica no botão “Salvar Como”
d. Um menu é exibido.
e. Usuário clica no botão “JSON”.

4.1.11 Selecionar uma sala

1 Objetivos

Permitir ao usuário acessar uma sala de sessão colaborativa.

2 Sumário

Permite ao usuário entrar em uma sala de sessão colaborativa a partir de uma chave. Gerar uma chave de acesso à sessão criada.

3 Atores

Qualquer entidade que quer ser um Host de uma sessão.

4 Fluxo normal

a. Usuário clica no botão “Gerenciador de sessões”;
b. O modal “Gerenciador de sessões” é exibido;
c. Usuário clica no botão “Iniciar nova sessão”;
d. É exibido no modal a chave de acesso a sessão criada.

5 Fluxos alternativo

a. A criação da sessão falha;
b. O usuário já está em uma sessão (ou como host ou como guest);

4.1.12 Terminar um SGBD

1 Objetivos

Permitir ao usuário que termina a execução de um SGBD.

2 Sumário

O usuário pode terminar a execução de um SGBD, de forma que não será mais possível executar SQL Scripts.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Sistema Gerenciador de Banco de Dados”;
b. O modal “Sistema Gerenciador de Banco de Dados” é exibido;
c. Usuário clica no botão “Parar”;

d. A execução do SGBD é terminada e esse é liberado.

4.1.13 Verificar a presença de anomalias nas Tabelas

1 Objetivos

Permitir ao usuário verificar a presença de anomalias nas Tabelas.

2 Sumário

O usuário pode verificar a presença de anomalias nas Tabelas do Banco de Dados.

3 Atores

Usuário.

4 Fluxo normal

a. Usuário clica no botão “Buscador de Anomalias”;
b. Um modal é exibido;
c. Usuário seleciona uma tabela da lista de Tabelas;
d. Uma lista de hints (sugestões) é exibida, indicando quais colunas da Tabela tiveram linhas que sugerem dependências implícitas/funcionais.

4.2 Protótipos

Nas Figuras 3-16 estão os protótipos de alta fidelidade do sistema em desktop, que pode ser dividido em três aplicações e três serviços.

A primeira aplicação (Figuras 3-7) apresenta informações sobre o sistema, como as suas notas de versão, manuais, etc. A segunda aplicação (Figuras 8-9) é usado para a modelagem conceitual e lógica de banco de dados. A terceira aplicação (Figura 10) é um laboratório/workbench para execução de SQL Scripts.

O primeiro serviço (Figura 11) é uma ferramenta para buscar anomalias nos dados das Tabelas atuais do banco de dados. O segundo (Figura 12) é o gerenciador de sessões para ambientes de modelagens cooperativas. A terceira aplicação (Figura 13) é o gerenciador do SGBD, ao qual está embebido na memória.

Nas Figuras 14-16 tem-se a representação do sistema em mobile.

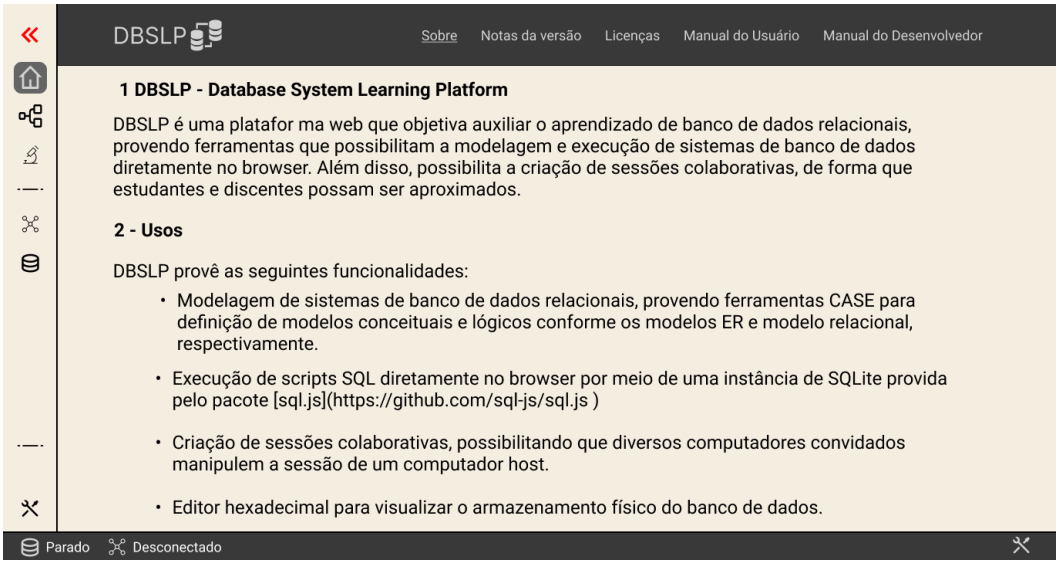


Figura 3 Desktop-Home-Page

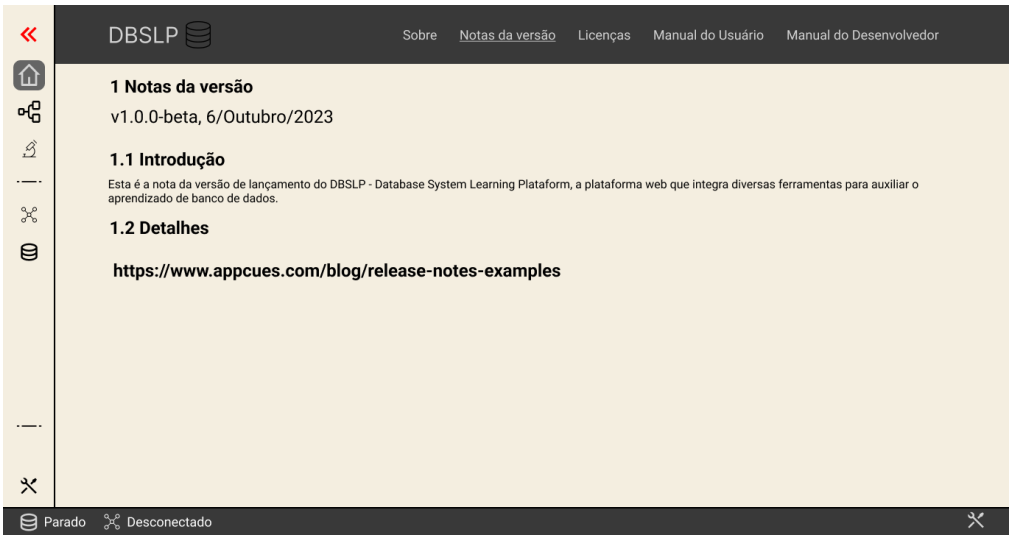


Figura 4 Desktop-Home-Page-2

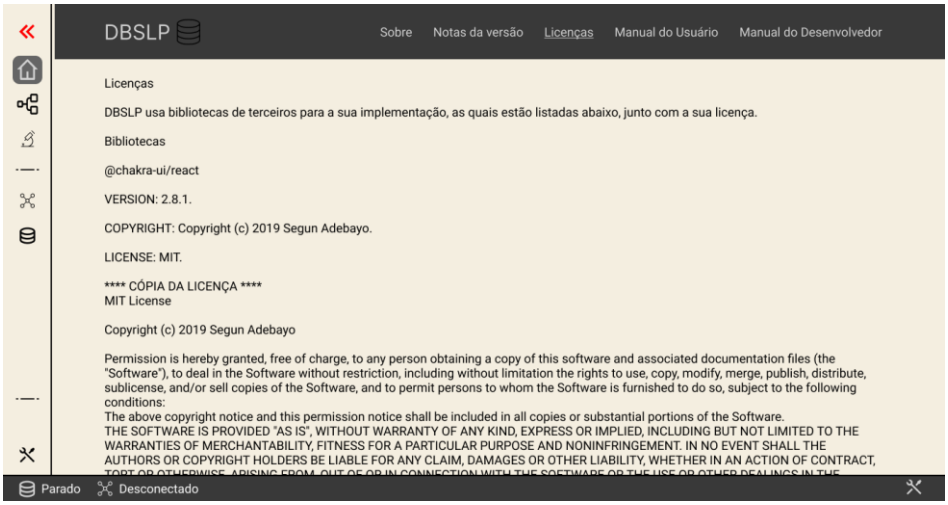


Figura 5 Desktop-Home-Page-3

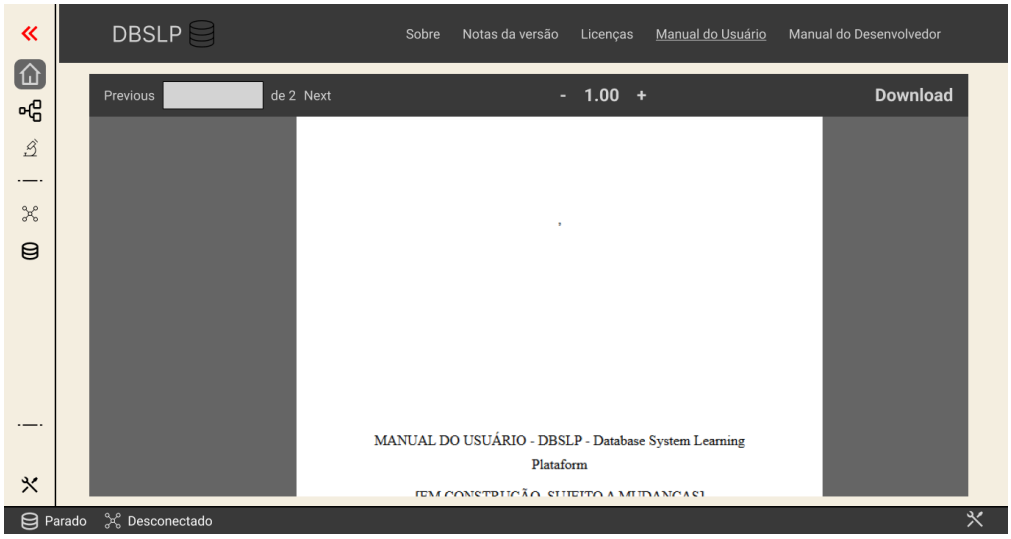


Figura 6 Desktop-Home-Page-4

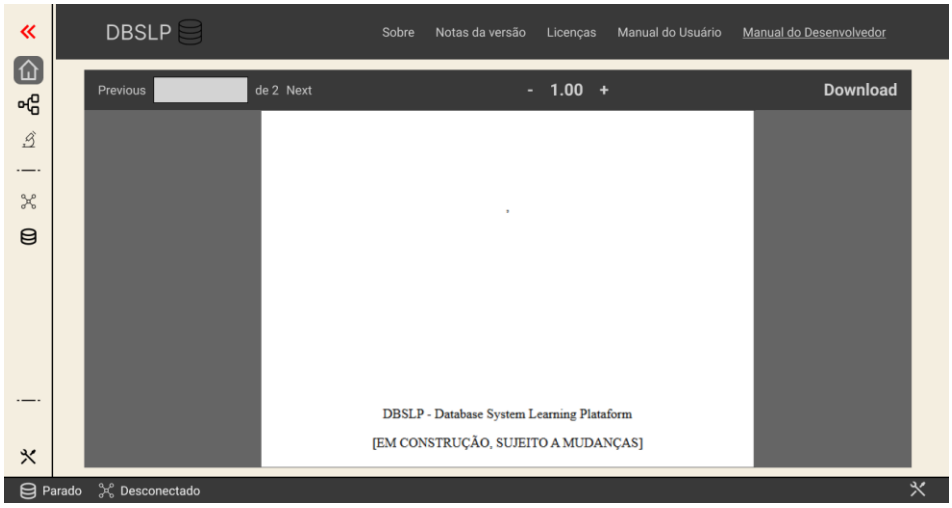


Figura 7 Desktop-Home-Page-5

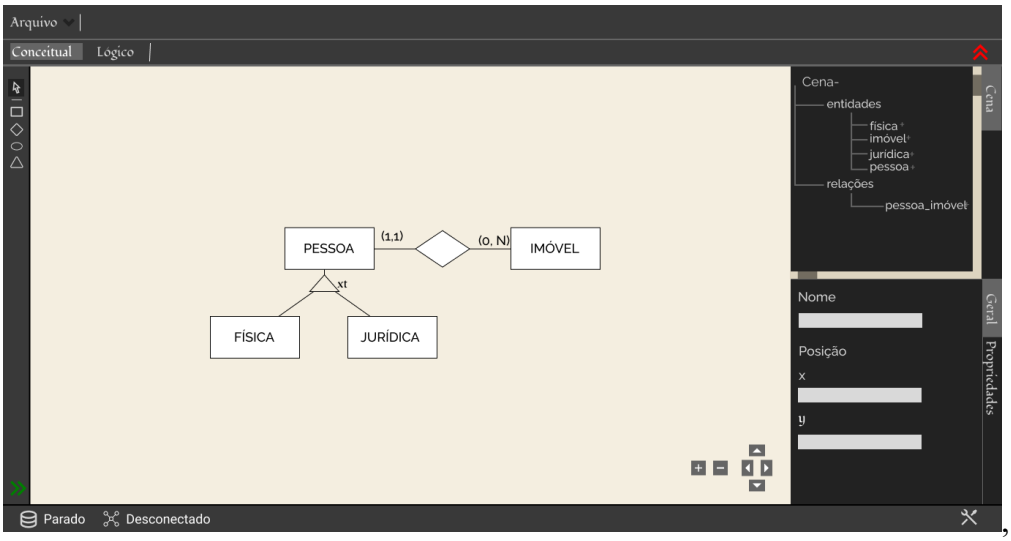


Figura 8 Desktop-Modelagem-Conceitual

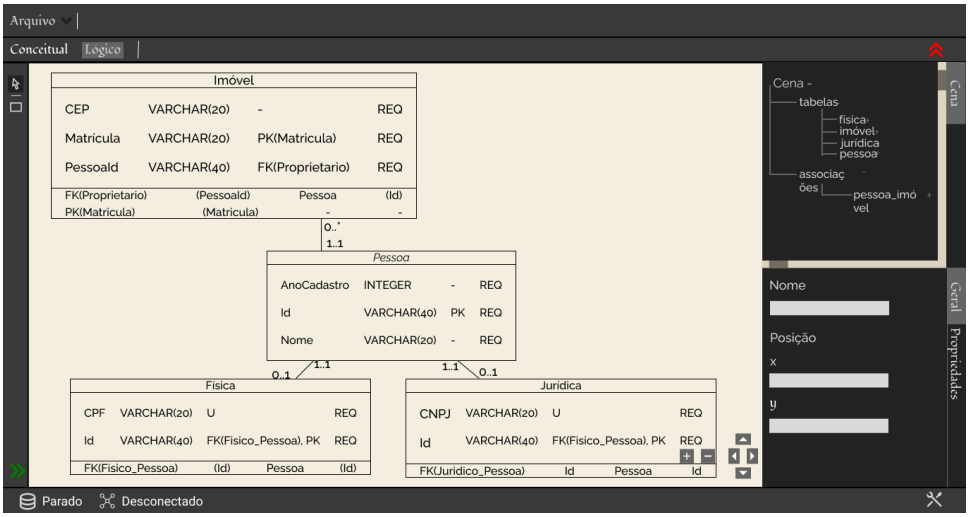


Figura 9 Desktop-Modelagem-Logica

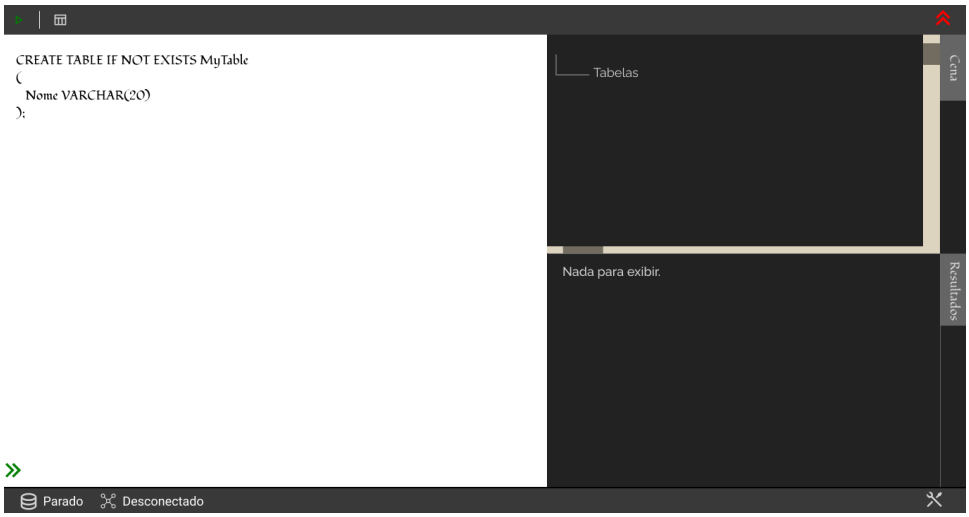


Figura 10 Desktop-SQL-Lab.

Figura 11 Desktop-Modal-Anomalias.

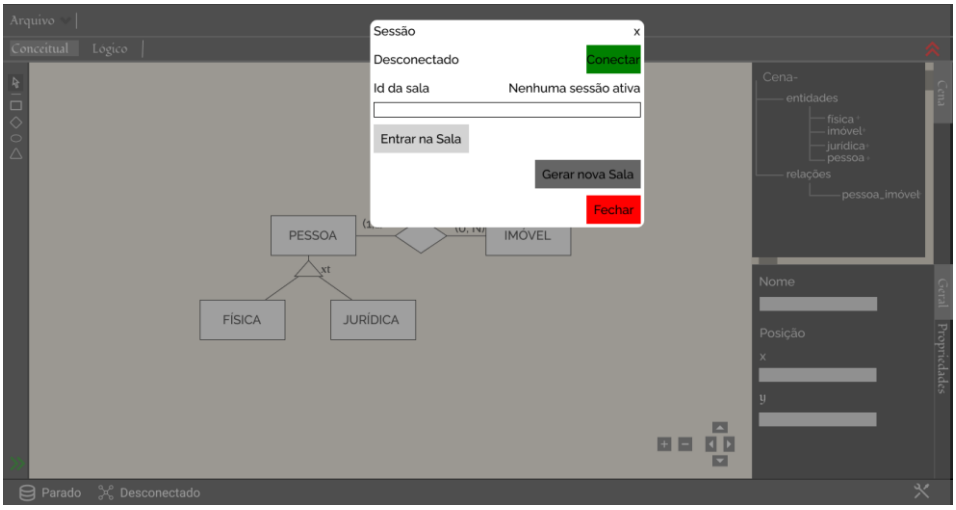


Figura 12 Desktop-Modal-Sessão.

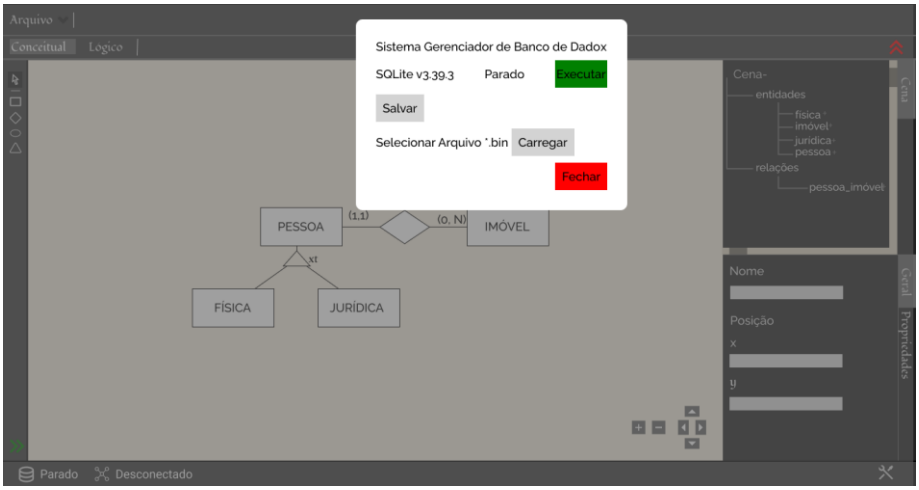


Figura 13 Desktop-Modal-SGBD.

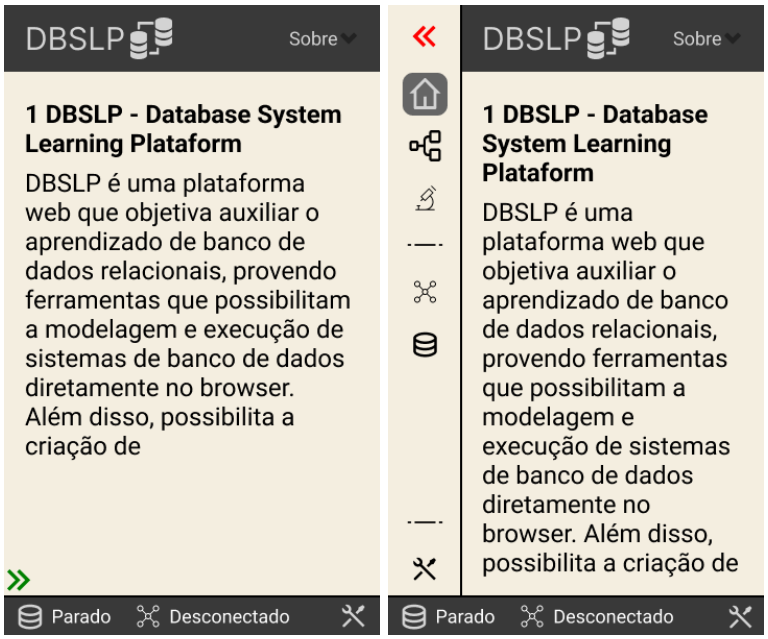


Figura 14 Mobile-Home

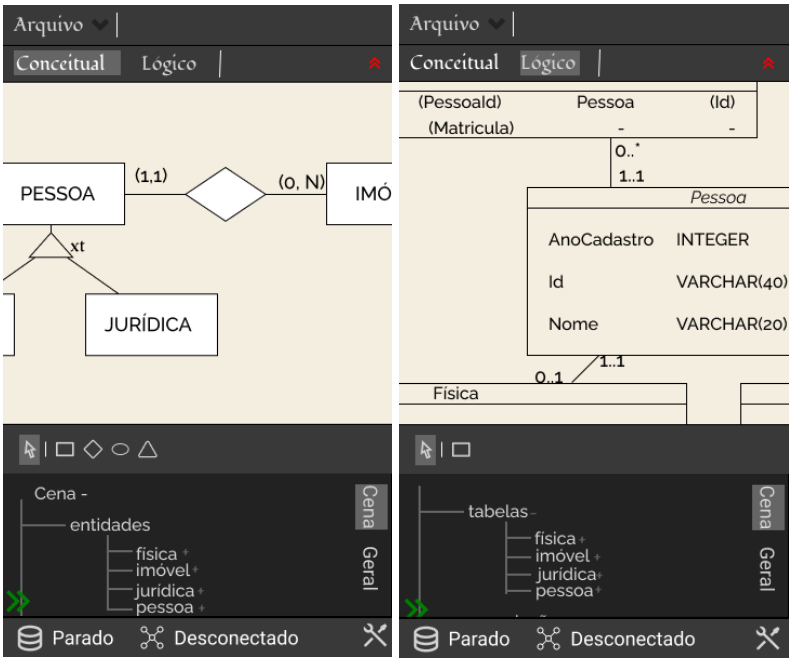


Figura 15 Mobile-Modelagem

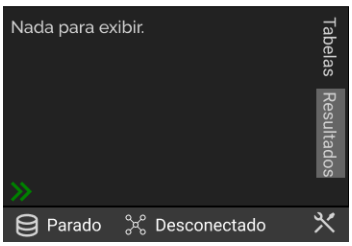


Figura 16 Mobile-SQL-lab.

5 Visão Lógica

Esta seção descreve a visão lógica do sistema, que consiste das decisões de nível mais baixo para o seu funcionamento.

Note que será descrito apenas a implementação da *front-end*, pois o servidor usado teve sua implementação elaborada pela equipe do yjs (https://docs.yjs.dev/). A implementação usada foi a *y-websocket,* versão 1.5.0, sob a licença MIT (https://github.com/yjs/y-websocket). Esse servidor é um servidor implementado no ecossistema Node.js baseado no protocolo websocket para sincronização entre as réplicas de documentos CRDTs.

5.1 Aplicação de modelagem

No ANEXO I (p.45) está o Diagrama de Classes. Nos Diagramas 2-5 estão as diversas partes desse diagrama comentadas.

As classes foram usadas para representar unidades de trabalho para as ferramentas de modelagem lógica e conceitual. Note que não foi usado algum método estático, seguindo um estilo “procedural”. Isso foi feito porque funções não são serializáveis como JSON e a sincronização dos documentos entre os usuários usa JSON, de forma que se optou pelo uso de métodos estáticos para as rotinas comuns.

No Diagrama 2 está representado as classes mais básicas. São aquelas que representam as propriedades gráficas (CanvasDetails, Vertex, Dimension e Rect) e a classe Cardinalidade, que é usada para representar a relação entre duas figuras/entidades e é usada tanto no modelo conceitual quanto no lógico.

No Diagrama 3 está representado as classes principais associadas à modelagem conceitual e lógica (Entidade, Relação, Atributo, Generalização, Tabela, Coluna, etc.), representando construções com valor semântico par ao usuário.

No Diagrama 4 está a classe Conexão e seus associados, que é usada para representar uma conexão entre duas figuras, que é um conjunto de segmentos, tendo um papel auxiliar.

No Diagrama 5 está a classe Store e seus associados, que são usados para representar o estado da modelagem em seu nível mais alto[2].

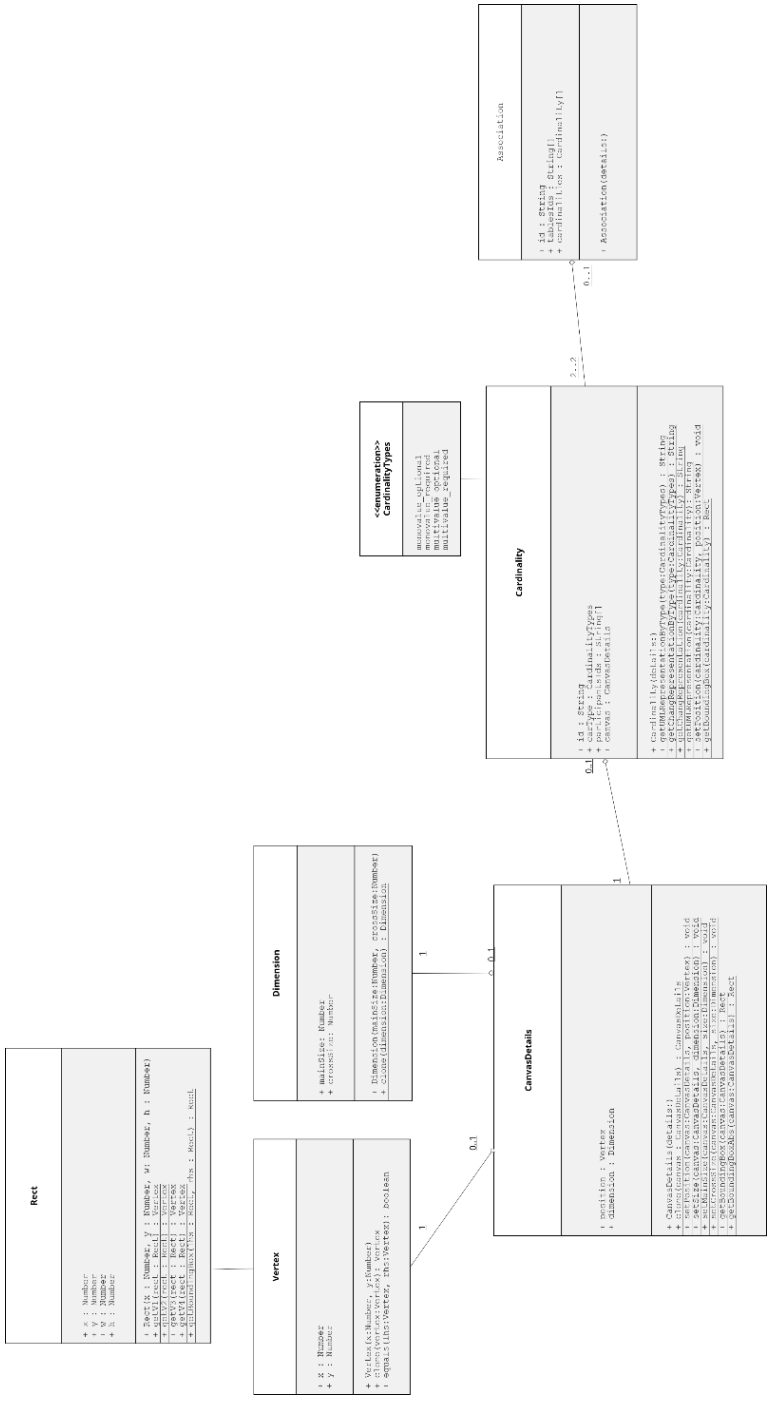


Diagrama 2 Diagramas de Classes Parte-1.

[2] Para a classe Store em si a implementação usou objeto com estrutura equivalente, mas não sendo uma instância de uma classe Store.

Diagrama 3 Diagramas de Classes Parte-2.

Diagrama 4 Diagramas de Classes Parte-3.

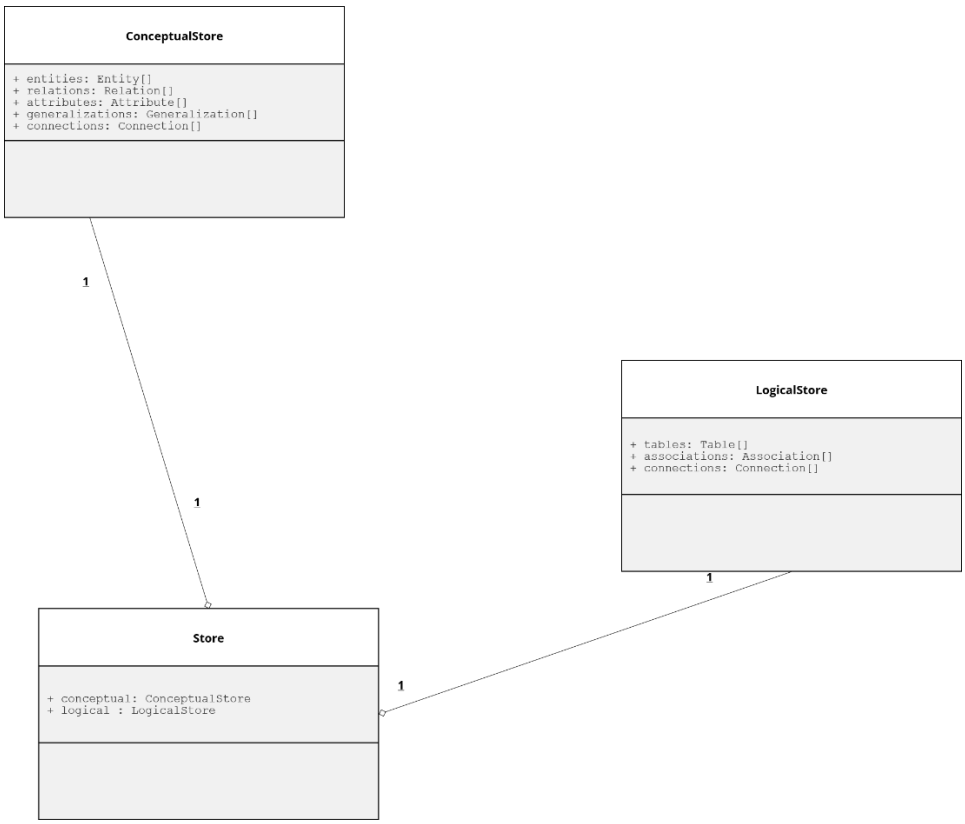


Diagrama 5 Diagramas de Classes Parte-4.

Para gerenciar o estado da interface das ferramentas de modelagem usou-se *statecharts* (*finite state machines)*, conforme representado no Diagrama 6. Essa foi usada para acoplar a resposta do sistema a interação do usuário com o viewport/canvas.

A máquina usada foi do tipo paralela consistindo de duas regiões com estado extendido compatilhado: (i) conceitual, para a modelagem conceitual (ii) lógica, para modelagem lógica. Nas Tabelas 1-3 estão todos os eventos possíveis e a sua descrição.

Os eventos da Tabela 1 são independentes da região, enquanto que os eventos da Tabela 2 e 3 são acoplados, respectivamente, à região conceitual e à região lógica.

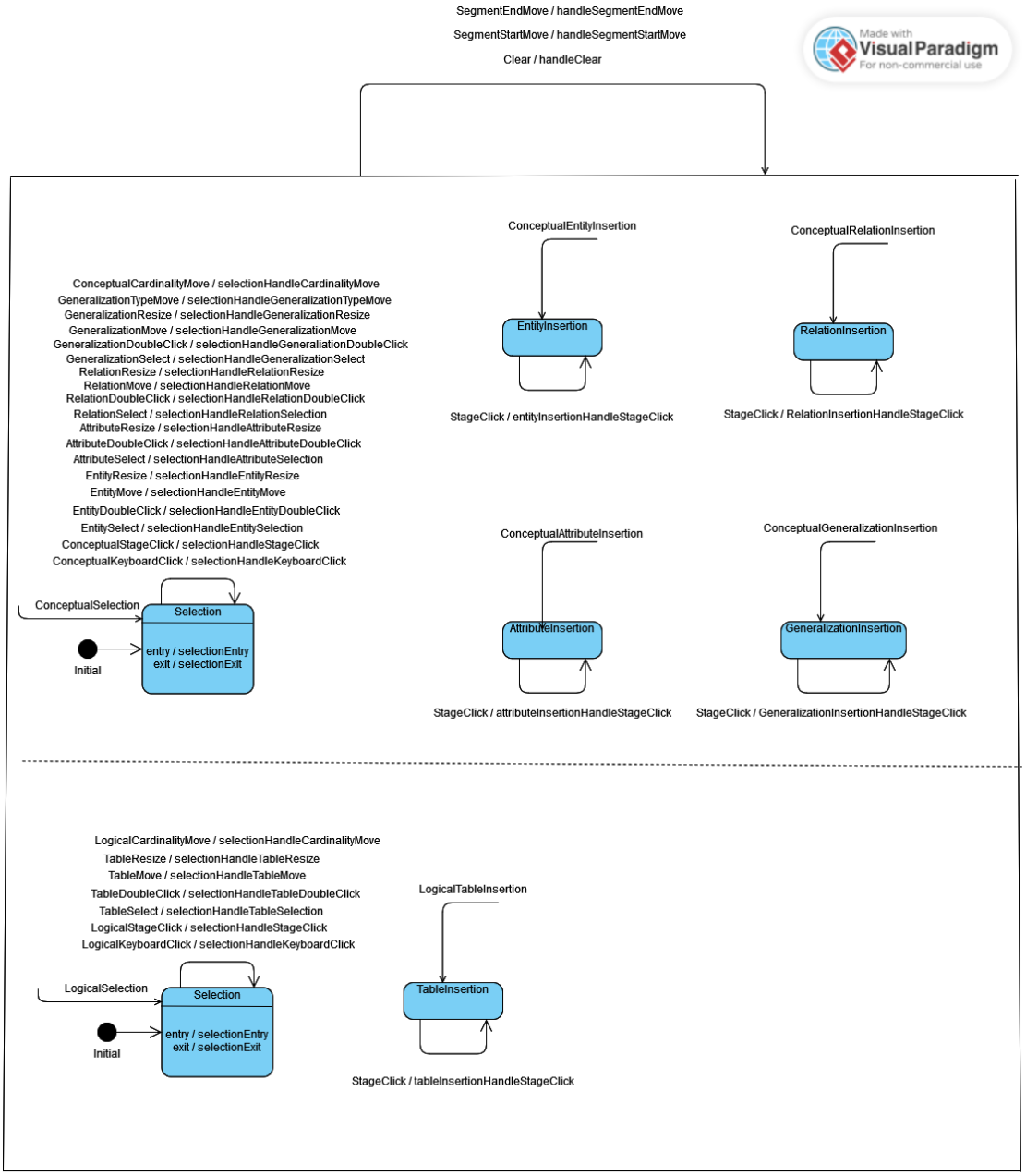


Diagrama 6 Diagrama de *finite state machine*.

Tabela 1 Eventos Gerais da Máquina de Estado.

Nome	Descrição
CONCEPTUAL_SELECTION	Modo de seleção para a modelagem conceitual.
CONCEPTUAL_ENTITY_INSERTION	Modo de inserção de entidade para a modelagem conceitual.
CONCEPTUAL_RELATION_INSERTION	Modo de inserção de relação para a modelagem conceitual.
CONCEPTUAL_ATTRIBUTE_INSERTION	Modo de inserção de atributo para a modelagem conceitual.
CONCEPTUAL_GENERALIZATION_INSERTION	Modo de inserção de generalização para a modelagem conceitual.
LOGICAl_SELECTION	Modo de seleção para a modelagem lógica.
LOGICAl_TABLE_INSERTION	Modo de inserção de tabela para a modelagem lógica.
CLEAR	Limpar todas as seleções do contexto.
SEGMENT_START_MOVE	Atualizar o vértice inicial de um segmento.
SEGMENT_END_MOVE	Atualizar o vértice final de um segmento.

Tabela 2 Eventos da Região conceitual da Máquina de Estado (Continua).

Nome	Descrição
ENTITY_SELECT	Usuário clicou em uma entidade.
ENTITY_DOUBLE_CLICK	Usuário clicou duplamente em uma entidade.
ENTITY_RESIZE	Usuário redimensionou uma entidade.
ENTITY_MOVE	Usuário moveu uma entidade.
ATTRIBUTE_SELECT	Usuário clicou em um atributo.
ATTRIBUTE_DOUBLE_CLICK	Usuário clicou duplamente em um atributo.
ATTRIBUTE_RESIZE	Usuário redimensionou um atributo.
ATTRIBUTE_MOVE	Usuário moveu um atributo.
RELATION_SELECT	Usuário clicou em uma relação.
RELATION_DOUBLE_CLICK	Usuário clicou duplamente em uma relação.
RELATION_RESIZE	Usuário redimensionou uma relação.
RELATION_MOVE	Usuário moveu uma relação.
CONCEPTUAL_CARDINALITY_MOVE	Usuário moveu uma cardinalidade no modelo conceitual.
GENERALIZATION_SELECT	Usuário clicou em uma generalização.
GENERALIZATION_DOUBLE_CLICK	Usuário clicou duplamente em uma generalização.

Tabela 2 Eventos da Região conceitual da Máquina de Estado (Conclusão).

Nome	Descrição
GENERALIZATION_RESIZE	Usuário redimensionou uma generalização.
GENERALIZATION_MOVE	Usuário moveu uma generalização.
GENERALIZATION_TYPE_MOVE	Usuário moveu o rótulo de título de uma generalização.
CONCEPTUAL_KEYBOARD_CLICK	Usuário pressionou o teclado no modelo conceitual.
CONCEPTUAL_STAGE_CLICK	Usuário clicou no Stage/Canvas no modelo conceitual.

Tabela 3 Eventos da Região lógica da Máquina de Estado (Continua).

Nome	Descrição
TABLE_SELECT	Usuário clicou em uma Tabela.
TABLE_DOUBLE_CLICK	Usuário clicou duplamente em uma Tabela.
TABLE_MOVE	Usuário moveu uma Tabela.
TABLE_RESIZE	Usuário redimensionou uma Tabela.

Tabela 3 Eventos da Região lógica da Máquina de Estado (Conclusão).

Nome	Descrição
LOGICAL_CARDINALITY_MOVE	Usuário moveu uma cardinalidade.
LOGICAL_KEYBOARD_CLICK	Usuário pressionou o teclado no modelo lógico.
LOGICAL_STAGE_CLICK	Usuário clicou no Stage/Canvas no modelo lógico.

Nos Diagramas 7 e 8 estão representados o workflow do sistema para alteração do estado da store/documento por meio de diagramas de atividades. Naquele, é a sequência comum para alteração por meio do viewport, em que o usuário seleciona um “modo” do sistema, interage com o viewport e o sistema reage de acordo com o seu estado, o qual foi representado o que insere uma nova entidade na posição de clique. Nesse, é a sequência da interação com um painel, que está acoplado a um estado e possibilita que o usuário altere o estado da modelagem, nesse caso o nome de uma Entidade.

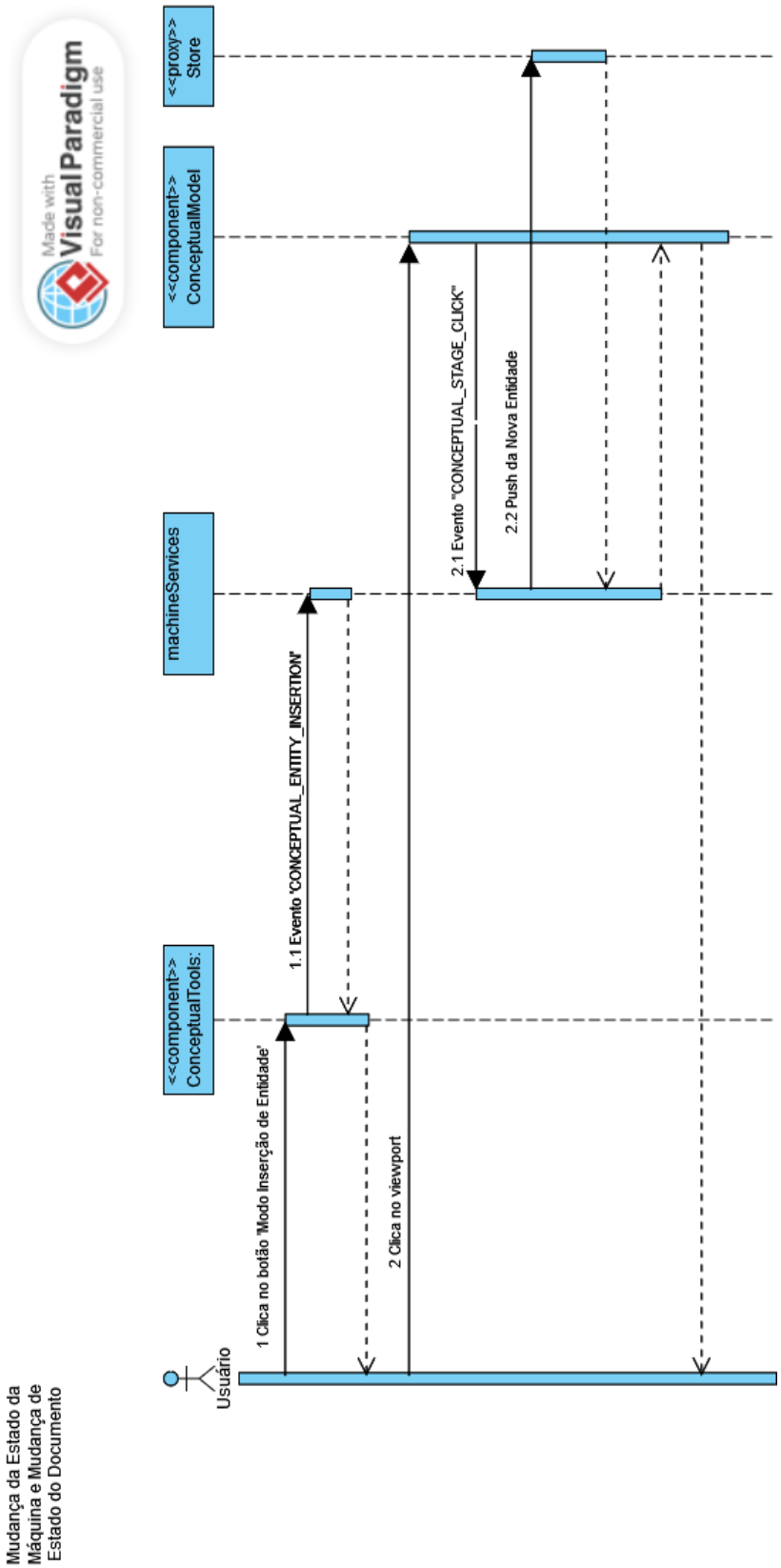


Diagrama 7 Diagrama de Sequência da Interação do usuário a partir do Viewport.

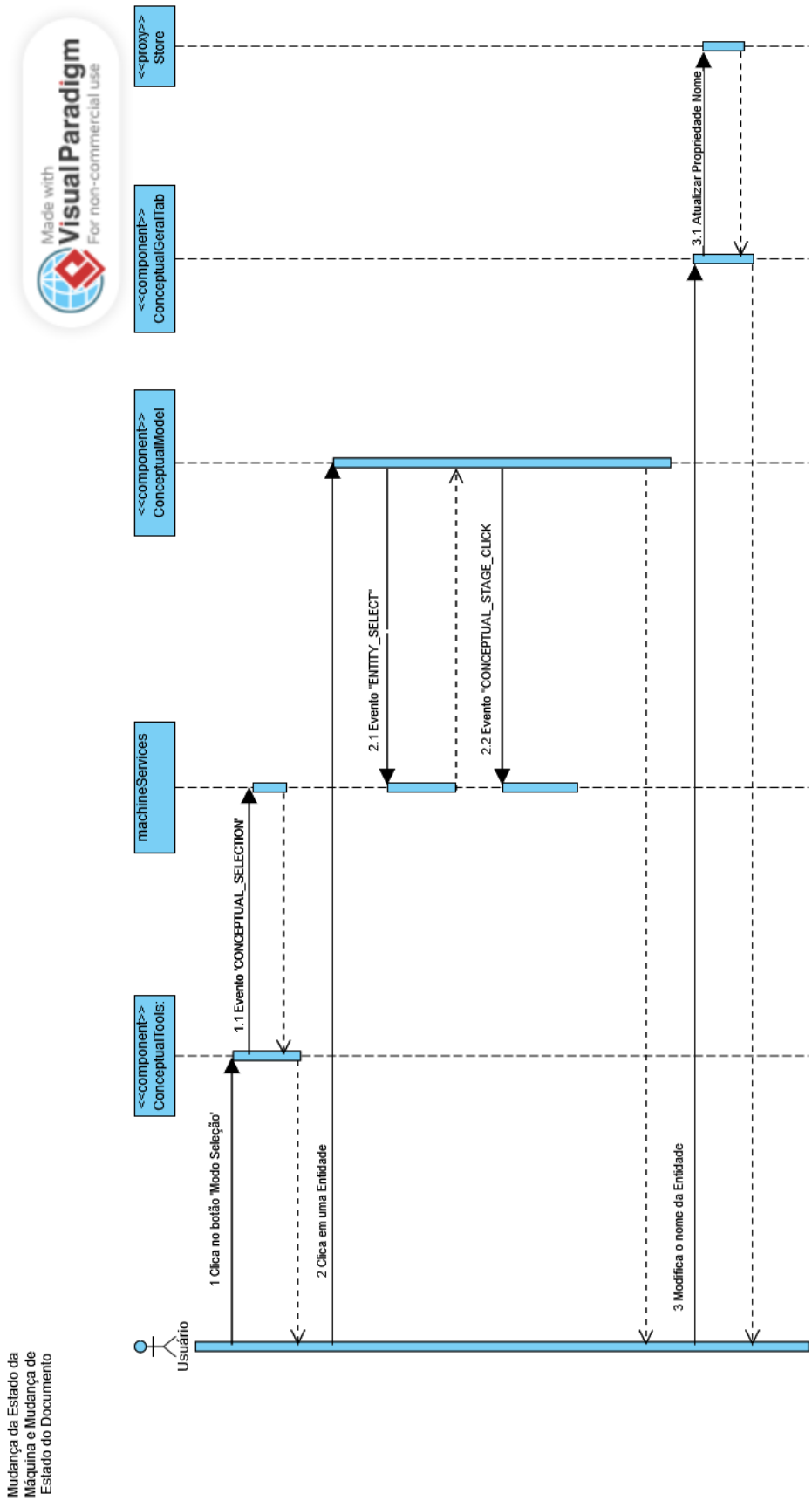


Diagrama 8 Diagrama de Sequência da Interação do usuário a partir de uma das Tabs.

5.2 Aplicação do laboratório SQL

No Diagrama 9 está representado a sequência de eventos para o usuário conseguir executar SQL Scripts, ao qual consiste da comunicação com o objeto que representa o SGBD, a criação de um objeto que representa os resultados, o seu preenchimento com os resultados e a sua definição como sendo o novo histórico dos resultados. O objeto resultado é definido como sendo o novo histórico quando a quantidade de resultados atual do histórico, quando adicionado a quantidade de novos resultados, é maior que um tamanho máximo de histórico arbitrário.

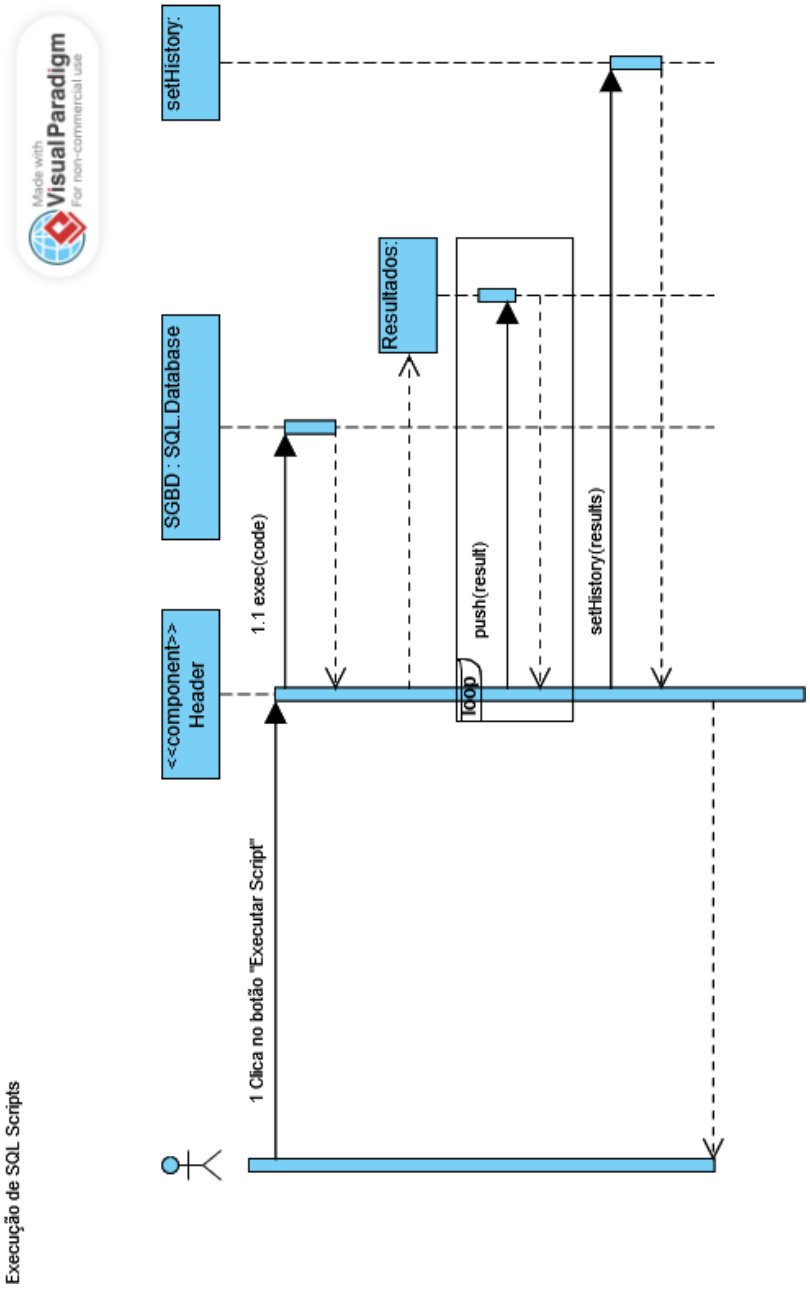


Diagrama 9 Diagrama de Sequência da execução de SQL Scripts.

6 Visão de Processos

Esta seção descreve o fluxo de sincronização entre processos dos nós que participam do sistema.

No Diagrama 10 está representado a sincronização entre dois documentos que colaboram na mesma sessão por meio de um servidor de broadcast.

Todos os nós (incluindo os servidores) tem uma réplica da Store, que representa a modelagem e é implementa como estruturas CRDTS[3]. Toda modificação do documento notifica o servidor de broadcast, que se comunica com o dispositivo do usuário por meio do protocolo *websocket*. Esse servidor, então, propagada essa modificação aos outros nós[4].

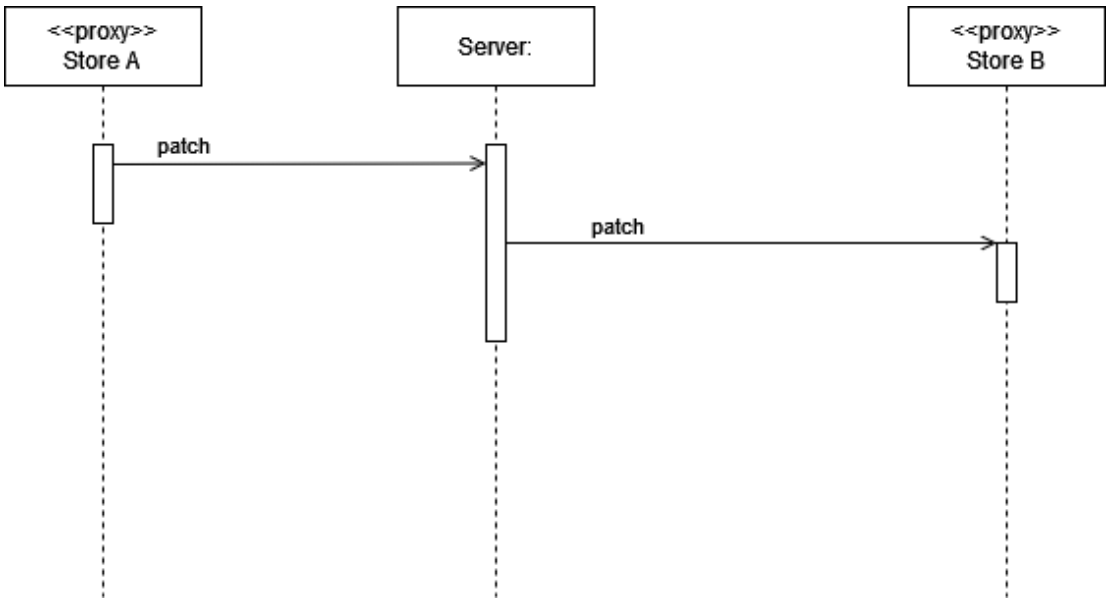


Diagrama 10 Diagrama de Processos

[3] A implementação usou a biblioteca SyncedStore (https://syncedstore.org), que internamente usa a biblioteca Yjs para a implementação de estruturas CRDTS (https://docs.yjs.dev/).
[4] Lembre-se que esse servidor foi implementado pela equipe do yjs como *y-websocket*.

7 Visão de Implementação

Esta seção descreve a organização de alto nível do sistema.

No Diagrama 11 está a hierarquia de pacotes da implementação, que representa a organização do sistema de *front-end*.

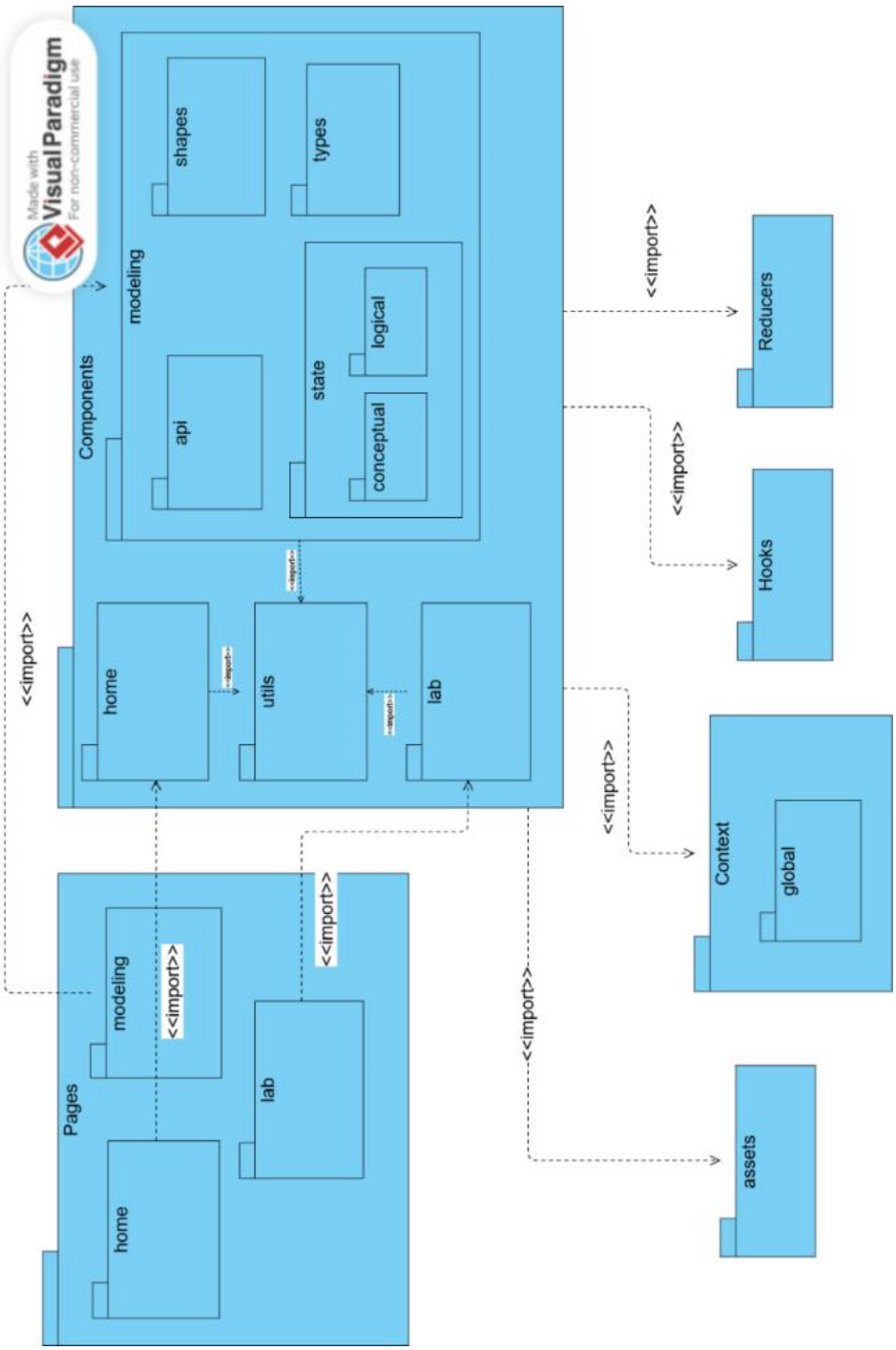


Diagrama 11 Diagrama de Pacotes.

8 Visão de Implantação

Esta seção descreve a topologia do sistema.

No Diagrama 12 está representada a topologia dos dispositivos que participam do sistema.

O sistema consiste de um servidor para prover os artefatos estáticos html/css/js (Application Server), que se comunica com o dispositivo do usuário por meio do protocolo *http*, e um servidor de broadcast para distribuir modificações em documentos de uma mesma sala, que se comunica com os dispositivos dos usuários por meio do protocolo *websocket*.

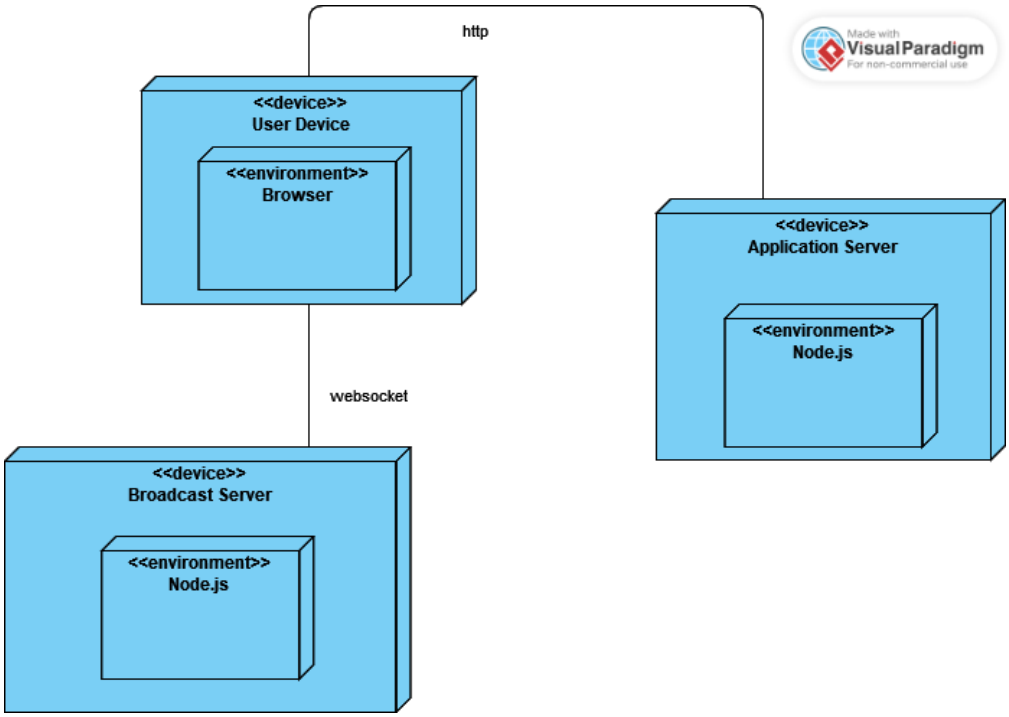


Diagrama 12. Diagrama de Implantação.

9 Licenças

Esta seção descreve as dependências de terceiros usados pelo sistema, especificando a sua versão e a sua justificativa, ou uso, no sistema.

Nas Tabelas 4-5 está apresentado as licenças dos pacotes ou dependências usadas no desenvolvimento do sistema.

Tabela 4 Licenças da front-end (Continua)

Nome	Versão	Licença	Justificativa
@chakra-ui/react	2.8.1	MIT	Estilização e componentes pré-elaborados
@emotion/react	11.11.1	MIT	Dependência de **@chakra-ui/react**
@emotion/styled	11.11.0	MIT	Dependência de **@chakra-ui/react**
@syncedstore/core	0.6.0	MIT	Gerenciar estruturas CRDTS para sessões colaborativas
@syncedstore/react	0.6.0	MIT	Adaptação de **@syncedstore/core** para o ecossistema React
@xstate/react	3.2.2	MIT	Adaptação do **xstate** para o ecossistema React
file-saver	2.0.5	MIT	API de download de arquivos no browser
framer-motion	10.16.4	MIT	Dependência de **@chakra-ui/react**

Tabela 4 Licenças da front-end (Continua)

Nome	Versão	Licença	Justificativa
line-intersect	3.0.0	MIT	API para detecção de intersecção de segmentos de Reta
mathjs	11.11.2	APACHE LICENSE 2.0	API para operações matemáticas
rc-tree	5.8.0	MIT	Biblioteca para React para criação de árvores/hierarquias
react	18.2.0	MIT	Biblioteca para CSR (Client Side Rendering)
react-custom-scrollbars-2	4.5.0	MIT	Biblioteca para React para barra de scrolls.
react-dom	18.2.0	MIT	Biblioteca de CSR (Client Side Rendering) para Browser sobre o **react**
react-icons	4.11.0	MIT	Biblioteca para React que representam ícones
react-konva	18.2.10	MIT	Biblioteca para React que abstrai o Canvas em computação 2D.
react-markdown	9.0.0	MIT	Biblioteca para React para interpretação e exibição de Markdown
react-pdf	7.5.0	MIT	Biblioteca para React para interpretação e exibição de PDF

Tabela 4 Licenças da front-end (Conclusão)

Nome	Versão	Licença	Justificativa
react-router-dom	6.16.0	MIT	Biblioteca para React para Client Side Routing
react-simple-code-editor	0.13.1	MIT	Biblioteca para React para editor de códigos
sql-highlight	4.4.0	MIT	API para realçar as diversas partes de scripts SQL
unique-names-generator	4.7.1	MIT	API para gerar nomes de forma pseudo-aleatória
xstate	4.38.2	MIT	API para representar *statecharts* (*finite state machines*)
y-websocket	1.5.0	MIT	API usada para gerenciar a conexão entre os dispositivos de sessões colaborativas
sql.js	1.8.0	MIT	API usada para criar e gerenciar instância de SQLite *in-memory*

Tabela 5 Licenças da back-end (Continua)

Nome	Versão	Licença	Justificativa
y-websocket	1.5.0	MIT	Implementação base para servidores baseados na biblioteca yjs.

10 Testes

Esta seção especifica como a aplicação deve ser testada.

As dependências de desenvolvimento para teste foram o pacote testing-library (https://www.npmjs.com/package/@testing-library/react) em conjunto com o pacote vitest (https://www.npmjs.com/package/vitest), que possibilitam testes unitários e de funcionalidade do sistema.

As APIs web ou de algoritmos devem ser testadas a partir de testes unitários. Os componentes React devem ser testados por meio de testes de funcionalidade[5].

[5] Implementou-se somente os testes das APIs de algoritmos, contudo.

11 Problemas conhecidos

Esta seção discorre sobre problemas conhecidos da versão atual da implementação do sistema.

- As transformações podem ter comportamento não-especificado;

- As conexões consistem de apenas um segmento;

- Os atalhos de teclado não foram implementados;

- A seleção do texto do visualizador de PDF não está funcionando corretamente;

- O indicador da rota atual no aplicativo ‘Home’ deixa de ser indicado quando não se está no path ‘/’.

- O canvas/viewport da Modelagem não se adequa à redimensionamentos;

- A aplicação de modelagem está com atraso/lag nas respostas de interação do usuário, como atrasos em mudanças de posições e translações quando o usuário interage com o canvas/viewport;

REFERÊNCIAS

BRUSILOVSKY, Pete *et al*. **Learning SQL Programming with Interactive Tools: From Integration to Personalization**. *ACM Transactions on Computing Education*, v9 n4 Article 19 Jan 2010.

HAMZAH, Muhammad Luthfi *et al*. **A review of increasing teaching and learning database subjects in computer Science**. Revista Espacios, Vol. 40 (Number 26), 2019, p.6.

HU, Chenglie. **An Introduction to Software Design** - Concepts, Principles, Methodologies, and Techniques. Springer, 2023.

MUHAMMAD, Ishaq *et al*. **Advances in database systems education: Methods, tools, curricula, and way forward**. Revista Springer, Education and Information Technologies, 2023.

POLETTO, Alex Sandro Romeo de Souza; SANTOS, Erik Correa; JÚNIOR, Jorge Rady de Almeida. **Avaliação do grau de dificuldade do aprendizado da modelagem de banco de dados**. COBENGE, XLI Congresso Brasileiro de Educação em Engenharia, 2013.

ANEXOS

ANEXO I – Diagrama de Classes

Diagrama 13 Diagrama de Classes