Documentação Técnica de Projeto

<TOINTO/2023-2024>

**Histórico de Alterações**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 22/03/2023 | Versão 1 | Criação da documentação | Natália Samogim |
| 24/03/2023 | Versão 2 | Modificação do documento | Natália Samogim |
| 07/06/2023 | Versão 3 | Acréscimo ao documento | Natália Samogim |
| 20/09/2023 | Versão 4 | Acréscimo ao documento | Júlia Dias |
| 17/11/2023 | Versão 5 | Acréscimo ao documento | Júlia Dias |
| 15/12/2023 | Versão 6 | Acréscimo ao documento | Luis Otávio |
| 05/03/2024 | Versão 7 | Acréscimo ao documento | Maria Luiza Poli |
| 26/03/2024 | Versão 8 | Acréscimo ao documento | Natália Samogim |
| 09/05/2024 | Versão 9 | Acréscimo ao documento | Laura Figueiredo |
| 13/06/2024 | Versão 10 | Acréscimo ao documento | Marília Bellini |

Sumário:

[1. Introdução 5](#_Toc169181571)

[2. Objetivo do Projeto 5](#_Toc169181572)

[3. Escopo 5](#_Toc169181573)

[4. Metodologia de Desenvolvimento de Sistemas 5](#_Toc169181574)

[5. Etapas e Time 6](#_Toc169181575)

[6. Cronograma 7](#_Toc169181576)

[7. Gestão de Recursos 12](#_Toc169181577)

[7.1 Definição da Equipe do Projeto 12](#_Toc169181578)

[7.2 Recursos Utilizados 15](#_Toc169181579)

[8. Stakeholders 16](#_Toc169181580)

[9. Riscos 18](#_Toc169181581)

10. Feedbacks............................................................................................................................17

11. Atualizações Feitas..............................................................................................................17

12. Atualizações a fazer.............................................................................................................18

13. Fluxo de Navegação.............................................................................................................19

14. Estrutura de Dados..............................................................................................................19

15. Tabela de Relacionamento...................................................................................................23

16. Banco de Dados...................................................................................................................24

16.1- Visão Geral das Tabelas do Banco........................................................................24

16.2- Visão da Estrutura da Tabela “cadastro”..............................................................25

16.3- Visão da Estrutura e Relação da Tabela “compras”.............................................25

16.4- Visão da Estrutura e Relação da Tabela “compromissos”....................................26

16.5- Visão da Estrutura e Relação da Tabela “dados\_pag”...........................................27

16.6- Visão da Estrutura e Relação da Tabela “historico”..............................................28

16.7- Visão da Estrutura da Tabela “opiniao”................................................................29

16.8- Visão da Estrutura da Tabela “planos”..................................................................30

17. Documentação do MCU......................................................................................................30

17.1 Lista de Requisitos.................................................................................................30

17.2 Atores ....................................................................................................................32

17.3 Fluxo principal - Lading Page................................................................................34

17.4 Fluxo Principal - calendário...................................................................................35

# Introdução

Este documento fornece uma apresentação técnica do projeto Tointo, sem se aprofundar muito nos detalhes da interação do usuário com o site e a interação da equipe, já que o último será realizado ao longo do projeto. Nele são apresentados os **objetivos do projeto**, as etapas de produção, como o **escopo**, a **metodologia de desenvolvimento**, as **etapas e time e o cronograma**. Também conterá a **gestão de recursos, os stakeholders e os possíveis riscos** que podem surgir ao longo do projeto, além da estrutura de dados do projeto e **tabela de relacionamento** das tabelas das funcionalidades do TOINTO no Banco de Dados

# Objetivo do Projeto

O projeto do site Tointo surgiu com o objetivo de ajudar e facilitar a organização da rotina, do trabalho e o cotidiano dos usuários de uma forma prática e funcional, pensado sempre na melhor experiência e interação deles com o produto.

# Escopo

O escopo do projeto da lista de tarefas Tointo, consiste na realização da criação do wireframe, prototipado em HTML, CSS, Python, React + vite com expo, React Bootstrap e React Native com o expo, com a criação da landing page (com página de login e cadastro, o home, os recursos, planos, sobre a empresa e as redes sociais) e do desenvolvimento do protótipo da página principal da lista de tarefas, tudo na paleta de cor roxa e azul, para remeter a realeza e poder.

A aplicação Tointo consiste em um calendário seguindo o modelo do wireframe, feito no framework React Bootstrap, além do React vite, do Python e do React Native com expo, que possui o menu que conta com o hoje, semana e importante, junto ao calendário que contém o To do List e traz em anexo o compromisso. Além disso, acima do calendário abrange o mês e o ano, podendo modificá-lo através dos botões de seta, e o ícone do perfil que contém o nome de usuário, email, botão para modificar cadastro, para modificar pagamento (escolha forma de pagamento e adicionar dados do cartão), modificar plano, de excluir conta e botão de sair.

# Metodologia de Desenvolvimento de Sistemas

Para o desenvolvimento deste projeto, foi utilizado o método Scrum, desta forma cada um da equipe recebeu as seguintes funções nas Sprints 1, 2 e 3:

* Natália Ap. Samogim: recebeu a função de PO;
* Júlia Dias Lara: recebeu a função de Scrum Master;
* Laura F. Figueiredo, Luis Otávio S. Batista e Marília M. Bellini: receberam a função de Desenvolvedores.
* Maria Luiza G. Poli: recebeu a função de arquiteta.

Na Sprint 4 e 5 cada membro da equipe recebeu as seguintes funções:

* Júlia Dias Lara: recebeu a função de PO;
* Maria Luiza Goulart Poli: recebeu a função de Agilista;
* Laura F. Figueiredo, Luis Otávio S. Batista, Marília M. Bellini e Natália Ap. Samogim: receberam a função de Desenvolvedores.

Na Sprint 6 cada membro da equipe recebeu as seguintes funções:

* Luis Otávio S. Batista: recebeu a função de PO;
* Marília M. Bellini: recebeu a função de Agilista;
* Laura F. Figueiredo, Natália Ap. Samogim, Júlia Dias Lara e Maria Luiza Goulart Poli: receberam a função de Desenvolvedores.

Na Sprint 7 cada membro da equipe recebeu as seguintes funções:

* Maria Luiza Goulart Poli: recebeu a função de PO;
* Laura F. Figueiredo: recebeu a função de Agilista;
* Marília M. Bellini, Natália Ap. Samogim, Júlia Dias Lara e Luis Otávio S. Batista: receberam a função de Desenvolvedores.

Na Sprint 8 cada membro da equipe recebeu as seguintes funções:

* Laura F. Figueiredo: recebeu a função de PO;
* Natália Ap. Samogim: recebeu a função de Agilista;
* Júlia Dias Lara, Marília M. Bellini, Maria Luiza Goulart Poli e Luis Otávio S. Batista: receberam a função de Desenvolvedores.

Na Sprint 9 cada membro da equipe recebeu as seguintes funções:

* Marília M. Bellini: recebeu a função de PO;
* Maria Luiza Goulart Poli: recebeu a função de Agilista;
* Júlia Dias Lara, Laura F. Figueiredo e Natália Ap. Samogim: receberam a função de Desenvolvedores.

# Etapas e Time

|  |  |  |
| --- | --- | --- |
| Sprint | Período | Objetivos |
| 1 | 120 horas | Entregar a Landing Page |
| 2 | 120 horas | Melhoria da aplicação TOINTO |
| 3 | 40 horas | Apresentar a Landing Page do TOINTO |
| 4 | 128 horas | Entregar a tela do To Do List e os Compromissos do TOINTO |
| 5 | 104 horas | Entregar a Landing Page integrada com o Calendário (com todas as rotas) |
| 6 | 56 horas | Entregar transferências de dados, validações do cadastro e do login e gravação dos dados |
| 7 | 112 horas | Aprendizagem e entrega da gravação de dados no Banco de Dados do projeto (cadastro inicialmente) |
| 8 | 104 horas | Entrega da aplicação no mobile |
| 9 | 88 horas | Entrega da parte funcional de nossa aplicação Mobile. |

# Cronograma

**Cronograma da parte de realização do TOINTO na Sprint 1 (25/01 a 15/03):**

* 25/01- Apresentação do curso;
* 27/01 – Aprendemos metodologias ágeis (Scrum e Kanban);
* 01/02 – Aprendemos a utilizar o Figma;
* 03/02- Começamos a aprender a estrutura de HTML;
* 08/02- Aprendemos a fazer o Wireframe no Figma;
* 10/02- Começamos a utilizar o CSS (display flex) e fazer o Wireframe e fizemos também a estrutura HTML;
* 15/02- Começamos a usar o GitHub e fazer a estrutura do CSS;
* 22/02 – Pesquisa sobre Front-end;
* 24/02 – Exercícios de Position e Display Grid;
* 01/03- Utilização de estrutura de nuvem para versionamento de código;
* 03/03- Aplicação dos layouts de alta fidelidade das páginas Home, Cadastro e Login.

**Cronograma da parte de realização do TOINTO na Sprint 2 (17/03 a 05/05):**

* 17/03- Terminamos o HTML e continuamos o CSS;
* 22/03- Fizemos ajustes no HTML, comentamos todos os códigos no HTML e encerramos o CSS e iniciamos a criação dos documentos de apresentação da Sprint 2;
* 24/03- Documentamos a parte funcional da LP em nível de usuário, a parte estrutural (HTML e CSS) em nível desenvolvedor, preparamos a apresentação para os Stakeholders (documentação técnica), os dados para review e dados para a retrospectiva;
* 28/03 e 31/03 – Produção da página de login em HTML e CSS;
* 31/03 – Produção da página principal em HTML e CSS;
* 12/04- Entrega final da Landing Page;
* 05/05 – Review e Retrospectiva da Sprint 2.

**Cronograma da parte de realização do TOINTO na Sprint 3 (10/05 a 16/06):**

* 10/05 - Desafios Back-end - exercícios de funções utilizando lógica de programação em Python;
* 17/05: Continuação de exercícios Back-end em Python;
* 26/05: Aprendemos e entregamos a atividade sobre a ferramenta 5W2H;
* 07/06: Entrega da documentação Sprint 2.

**Cronograma da parte de realização do TOINTO na Sprint 4 (26/07 a 20/09):**

* 26/07- Fizemos o wireframe do Hoje, Semana, Importante, Calendário, To Do List e Compromissos;
* 28/07 – Fizemos a estrutura de dados e formulário das funções;
* 02/08- Fizemos o Fluxo Principal, o Diagrama de Caso de Uso e a Lista de Requisitos;
* 04/08- Fizemos o Diagrama de Classes e arrumamos erros do Diagrama de Caso de Uso e Lista de Requisitos;
* 09/08- Concluímos o Diagrama de Classes e arrumamos erros da Landing Page;
* 11/08- Fizemos exercícios de Java Script;
* 16/08- Começamos a validação dos campos da nossa aplicação (login e cadastro);
* 18/08- Finalizamos a validação dos campos da nossa aplicação (Landing Page do TOINTO);
* 23/08- Instalamos o node e começamos o conteúdo sobre React;
* 25/08- Finalizamos a página dos jogos utilizando o React e iniciamos a página do Instagram, que foi feita individualmente;
* 30/08- Finalizamos a aplicação do Instagram (individual);
* 01/09 – Entrega individual da tela do Instagram, planejamento da página do To do List, correção na validação de alguns campos de cadastro;
* 06/09 – Planejamento da página do To Do list, inicialização da página To do List e compromissos web (react) e grupo;
* 13/09 – Início da página web do To Do List e compromissos pelo React Bootstrap individual;
* 15/09 – Finalização da página do To do List e compromissos individual e inicialização do To do List e compromissos para o TOINTO;
* 20/09- Fizemos as documentaçãos, comentamos os codigos da nossa lista de tarefas tointo;
* 22/09 – Planejamento da Sprint 5 e Review e Retrospectiva da Sprint 4.

**Cronograma da parte de realização do TOINTO na Sprint 5 (27/09 a 17/11)**

* 27/09 – Revisão dos planejamentos do Kanban do projeto da Sprint 5 e início da parte teórica sobre rotas (exercícios individuais também);
* 29/09 – Correção dos códigos necessários das páginas de Landing Page e da página principal do calendário;
* 04/10 – Integração do Froentend com o Backend;
* 06/10 – Exercício para entregar sobre integração do Frontend com o Backend;
* 11/10 – Integração da Landing Page com o React;
* 18/10 – Integracão da Landing Page com o React;
* 20/10 – Integração da Landing Page com o React;
* 25/10 – Término das rotas e css da Landing Page e início da rota e css do calendário;
* 27/10 – Continuação das rotas e css do Calendário;
* 01/11– Término das rotas e do css do Calendário e início de exercícios com Inteligência Artificial;
* 08/11 – Correção das documentações do projeto;
* 10/11 – Finalização das correções do projeto;
* 17/11 – Apresentação do site e termino da sprint.

**Cronograma da parte de realização do TOINTO na Sprint 6 (22/11 a 15/12)**

* 22/11 – Integração do Frontend com o Backend por meio do Json;
* 24/11 – Backend, tratamento de dados via API e orientação ao React;
* 29/11 – Validação Backend do cadastro;
* 01/12 – Tratamento da comunicação Backend e Frontend, validações do cadastro e login do TOINTO;
* 06/12 - Finalização da gravação dos dados, concluir o código;
* 13/12 – Ajustes do projeto e documentação;
* 15/12 – Entrega, apresentação das telas funcionais e o código.

**Cronograma da parte de realização do TOINTO na Sprint 7 (23/01 a 07/03)**

* 23/01 – Planning do projeto para 2024;
* 25/01 – Início do conteúdo sobre Banco de Dados;
* 30/01 – Continuação do conteúdo sobre Banco de Dados e exercícios;
* 01/02 – Início da Estrutura de Dados do TOINTO;
* 06/02 – Continuação da estrutura de dados do TOINTO, início da explicação sobre Normalização e exercícios sobre isso
* 08/02 – Continuação dos exercícios sobre Normalização (desenho das tabelas no Canva e exercícios no MySQL). Início dos exercícios sobre Select no BD;
* 13/02 – Não teve aula;
* 15/02 – Termino dos exercícios sobre Select no Banco de Dados e início do conteúdo sobre o comando Select e realização de exercícios sobre esse conteúdo.
* 20/02 – Término dos exercícios da aula passada e início de novos exercícios no Banco de Dados;
* 22/02 – Término dos exercícios no Banco de Dados, início da modelagem de dados do nosso projeto e realização das tabelas no Banco de Dados que criamos no Canva;
* 27/02 – Subimos o Banco de Dados do TOINTO no Github, começamos a fazer os inserts no Backend (Python);
* 29/02 – Fizemos códigos no Python (CRUD);
* 05/03 – Finalização do exercício sobre Insert, Update, Delete e início da gravação dos dados do Cadastro no Banco de Dados:
  + Validar entrada dados formulário;
  + Validar retorno gravação para o front;
  + Validar estrutura lista recebida pelo processamento;
  + Validar envio dos dados -lista- para o gravar banco;
  + Criar gravar banco;
  + Criar conexão;
  + Validar gravar banco;
  + Validar conexão;
  + Validar gravação no BD
* 07/03– Finalização do Insert e ajuste do Cadastro, finalização da Sprint, com Review e Retrospectiva da equipe e implementação do Update, Select e Delete na aplicação.

**Dias entre a Sprint 7 e 8 para concluir algumas partes do projeto:**

* 12/03 **–** Finalização da manipulação de dados (Update e Delete) e início dos Form Manipulação.
* 14/03–Inicio da Session do Login e novas páginas de pagamento (Adicionar dados do cartão, página de pix, Modificar dados do cartão e Modificar escolha de pagamento);
* 19/03 – Término do processamento do login, recuperação do cadastro, validação e criação de rotas;
* 21/03 – Término do select do login (aparecer o nome e-mail no Perfil), adicionar a foto ao perfil e término das validações dos dados do cartão.
* 26/03 – Tentamos fazer o Update e o Delete (não finalizou);
* 28/03 – Wireframe do TOINTO em mobile (login, calendário e To do list);

**Cronograma da parte de realização do TOINTO na Sprint 8 (26/03 a 09/05)**

* 02/04 – Prompts na IA e Select
* 04/04 – Termino do Select, Wireframe do cadastro e pagamento do mobile.
* 09/04 – Não teve aula.
* 11/04 – Preparação do ambiente mobile.
* 16/04 – Termino do ambiente mobile, inicio do app mobile.
* 18/04 – Termino do app mobile.
* 23/04 – Realizado um exemplo de tela de login e cadastro.
* 25/04 – Termino do exemplo de tela de login e cadastro e inicio do login e cadastro do TOINTO.
* 30/04 – Continuação páginas mobile TOINTO e início de rotas.
* 02/05 – Término dos exercícios de rotas.
* 07/05 – Inicio de rotas no TOINTO, e preparação para a apresentação de termino da sprint.
* 09/05 – Preparação para a apresentação e apresentação de termino da sprint.

**Cronograma da parte de realização do TOINTO na Sprint 9 (14/05 a 18/06)**

* 14/05 – Criação das páginas faltantes em nossa aplicação Mobile: página de perfil; modificar dados do usuário; hoje, semana e importantes; plano anual; formulário de adicionar dados do cartão; formulário para modificar dados do cartão; pagamento com o pix e modificar escolha de pagamento;
* 16/05 – Finalização das páginas faltantes em nossa aplicação;
* 21/05 – Início do conteúdo mobile em relação a página do calendário e sua produção em nossa aplicação oficial;
* 23/05 – Finalização da página do Calendário em nossa aplicação mobile;
* 28/05 – Início backend para página Login no Mobile e arrumar o botão excluir da página de perfil no Desktop;
* 30/05 – Criação da tela de modal em nossa aplicação mobile para excluir conta. Atualização de nossa aplicação no Desktop: Perfil (Modificar plano e deletar conta);
* 04/06 – Análise do projeto com o objetivo de prever possíveis impactos em nossa aplicação no futuro. Início da construção do CRUD de cadastro e compromissos no TOINTO mobile, e desenho das navegações dessa nossa aplicação;
* 06/06 – Continuação da construção do CRUD do cadastro e compromissos em nosso aplicativo mobile, início da produção para puxar a data na página de compromissos e todolist em nossa aplicação Desktop.
* 11/06 - Finalização do CRUD do cadastro e início do CRUD do cartão em nossa aplicação mobile.
* 13/06 - Finalização das atividades faltantes da aula passada, revisão geral em toda a documentação já realizada durante os dias e preparação da apresentação de nossa aplicação

# Gestão de Recursos

## Definição da Equipe do Projeto

Tabela com o mapeamento dos profissionais da equipe de projeto nas respectivas responsabilidades necessárias na Sprint 1.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe de projeto |
| Gerente de Projeto | Natália Ap. Samogim |
| PO | Natália Ap. Samogim |
| Scrum Master | Júlia Dias Lara |
| Desenvolvedor | Marília Bellini, Luís Otávio Batista e Laura Figueiredo |
| Arquiteto | Maria Luiza G. Poli |

**7.1.1 Alteração da equipe de projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 2.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Natália Ap. Samogim |
| PO | Natália Ap. Samogim |
| Scrum Master | Júlia Dias Lara |
| Desenvolvedor | Marília Bellini, Luís Otávio Batista e Laura Figueiredo |
| Arquiteto | Maria Luiza Goulart Poli |

**7.1.2 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 3.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Natália Ap. Samogim |
| PO | Natália Ap. Samogim |
| Scrum Master | Júlia Dias Lara |
| Desenvolvedor | Marília Bellini, Luís Otávio Batista e Laura Figueiredo |
| Arquiteto | Maria Luiza Goulart Poli |

**7.1.3 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 4 e Sprint 5.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Júlia Dias Lara |
| PO | Júlia Dias Lara |
| Agilista | Maria Luiza Goulart Poli |
| Desenvolvedor | Marília Bellini, Luís Otávio Batista, Laura Figueiredo e Natália Aparecida Samogim |

**7.1.4 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 6.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Luis Otávio Batista |
| PO | Luis Otávio Batista |
| Agilista | Marília Bellini |
| Desenvolvedor | Laura Figueiredo, Natália Aparecida Samogim, Júlia Dias Lara e Maria Luiza Goulart Poli |

**7.1.5 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 7.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Maria Luiza Goulart Poli |
| PO | Maria Luiza Goulart Poli |
| Agilista | Laura F. Figueiredo |
| Desenvolvedor | Luis Otávio Batista, Natália Aparecida Samogim, Júlia Dias Lara e Marília Bellini |

**7.1.6 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 8.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Laura F. Figueiredo |
| PO | Laura F. Figueiredo |
| Agilista | Natália Ap. Samogim |
| Desenvolvedor | Maria Luiza Goulart Poli, Júlia Dias Lara e Marília Bellini |

**7.1.7 Alteração da equipe do projeto**

Tabela com o mapeamento dos profissionais da equipe do projeto nas respectivas responsabilidades necessárias na Sprint 9.

|  |  |
| --- | --- |
| Responsabilidades | Profissionais da equipe do Projeto |
| Gerente do Projeto | Marília Bellini |
| PO | Marília Bellini |
| Agilista | Maria Luiza Goulart Poli |
| Desenvolvedor | Júlia Dias Lara, Laura F. Figueiredo e Natália Aparecida Samogim |

## Recursos Utilizados

Os recursos que foram utilizados ao longo do projeto são:

* JavaScript – ES2022
* Canva
* HTML5
* CSS3
* Freepick
* Python 3.11
* React - v18.2.0 com o vite
* React Bootstrap5 - v2.9.1
* GitHub Desktop – v3.11.0
* GitHub – v3.11.0
* Flaticon
* Bootstrap Icons – v2.9.1
* React-router-dom - v6.20.1
* Flask-CORS
* Flask
* MySQL/MariaDB
* Android Studio 2023.2.1
* React Native com Expo
* Emulador Nox

# Stakeholders

Tabela com o mapeamento da Sprint 1,2 e 3:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do Projeto | Sesi/Senai |
| Coordenador do Projeto | Natália Ap. Samogim |
| Diretores | Lucas Bueno e Junior Garcia |

Tabela com o mapeamento da Sprint 4 e 5:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do Projeto | Sesi/Senai |
| Coordenador do Projeto | Júlia Dias Lara |
| Diretores | Lucas Bueno e Junior Garcia |

Tabela com o mapeamento da Sprint 6:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do projeto | Sesi/Senai |
| Coordenador do projeto | Luis Otavio S. Batista |
| Diretores | Lucas Bueno e Junior Garcia |

Tabela com o mapeamento da Sprint 7:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do projeto | Sesi/Senai |
| Coordenador do projeto | Maria Luiza Goulart Poli |
| Diretores | Lucas Bueno e Junior Garcia |

Tabela com o mapeamento da Sprint 8:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do projeto | Sesi/Senai |
| Coordenador do projeto | Laura Francine Figueiredo |
| Diretores | Lucas Bueno e Junior Garcia |

Tabela com o mapeamento da Sprint 9:

|  |  |
| --- | --- |
| Stakeholder | Responsabilidade |
| Patrocinador do projeto | Sesi/Senai |
| Coordenador do projeto | Marília Bellini |
| Diretores | Lucas Bueno e Junior Garcia |

# Riscos

Os possíveis riscos que podem surgir ao longo do projeto são a utilização do GitHub Desktop e a do site (caso haja conflitos entre os documentos e outros problemas de push e pull), mudanças não previstas no projeto, a estruturação dos códigos, troca ou saída de membros, o atraso na aprendizagem/entrega dos participantes, o não cumprimento das atividades previstas e problemas que possam surgir na manipulação dos dados no Banco de Dados (como problemas de amarração, os relacionamentos das tabelas com as chaves, problemas com o where, delete e update feitos de forma errada que possam modificar o projeto inteiro), podemos sofrer falha no emulador e atrasar o processo.

1. **Feedbacks**

**Sprint 2 -** Na nossa apresentação sobre o TOINTO na escola, alguns que assistiram deram feedbacks, como arrumar o menu (deixá-lo fixo) e ter a opção de dar nota por estrelas também.

**Sprint 5 –** Trocar de lugar o calendário que antes estava na landing page e mudá-lo para o cadastre-se grátis e os assine agora dos planos.

**Sprint 8 –** Modificar no desktop o campo de opinião na landing page em nossa página, para que somente pessoas já cadastradas no aplicativo possam realizar alguma sugestão. Optamos por retirar esse campo, e futuramente iremos retirar também em nosso banco de dados. Além disso, também foi citado sobre modificar a escrita do botão “cadastre-se grátis” em nossa landing page para “cadastre-se já”, entretanto, é necessário a ideia ser discutida com o grupo para uma possível atualização no futuro.

1. **Atualizações Feitas**

As últimas atualizações feitas no projeto inicial, foi a mudança no logo (que agora ficou redonda), acréscimo de informações sobre o significado TOINTO no main e no footer da aplicação. Foi concluído também a funcionalidade do botão voltar das páginas de pagamento, que antes não estava funcionando.

Nas últimas atualizações, foi feito em nossa landing page, as validações no formulário (como por exemplo o campo do nome, senha e e-mail) e por último deixamos o nosso menu fixo. Foi feito pelo Framework React os a página do calendário, com Alterar Compromissos, To Do List e o Menu Lateral (com o Hoje, Semana e Importantes).

Nas últimas atualizações na **Sprint 4**, na Landing Page, foram os recados na senha do login, e no Calendário, precisamos separar o de lápis (editar o compromisso) e o lixo (excluir o compromisso) e deixar o título do compromisso.

As últimas atualizações da **Sprint 5** foram a integração da Landing Page com o Calendário, as rotas feitas na Landing Page e no calendário, arrumamos o cadastro na Landing Page, colocamos ele como uma página no Cadastre-se grátis e no assine agora de cada plano, fizemos o calendário (Ano, Mês e Semana) e o wireframe de esqueceu a senha.

As últimas atualizações na **Sprint 6** foram fazer a integração com o Json, as validações e gravações dos dados do cadastro e login.

As ultimas atualizações feitas na **Sprint 7** foram a integração e gravação dos dados da página de Cadastro com o Banco de Dados, ou seja, os dados inseridos no campo de Cadastro são gravados no banco.

As últimas atualizações feitas **entre a Sprint 7 e 8 (período para concluir algumas páginas do projeto)** foram a criação das páginas de adicionar e modificar dados do cartão (como o nome completo do titular, CPF, número do cartão, dada de vencimento e código de segurança) para efetuar o pagamento, página do pix (no qual aparece um QRCODE) para a realização do pagamento, modificar escolha de pagamento (no qual pode mudar a escolha de pix para cartão e vice-versa), modificação do perfil (no qual o usuário pode modificar a foto de perfil, nome, o e-mail e a senha), aparecer no perfil o nome e e-mail que usuário fez o login, páginas de Cadastro concluído e Cadastro modificado (para quando o usuário realizar o cadastro e quando modifica-lo, aparece essas páginas para informá-lo), além salvar os dados no localstorage, update e delete dos dados do cadastro e algumas mudanças na parte de CSS quando necessário (pois ao implementar rotas e outras páginas, teve que mudar algumas partes).

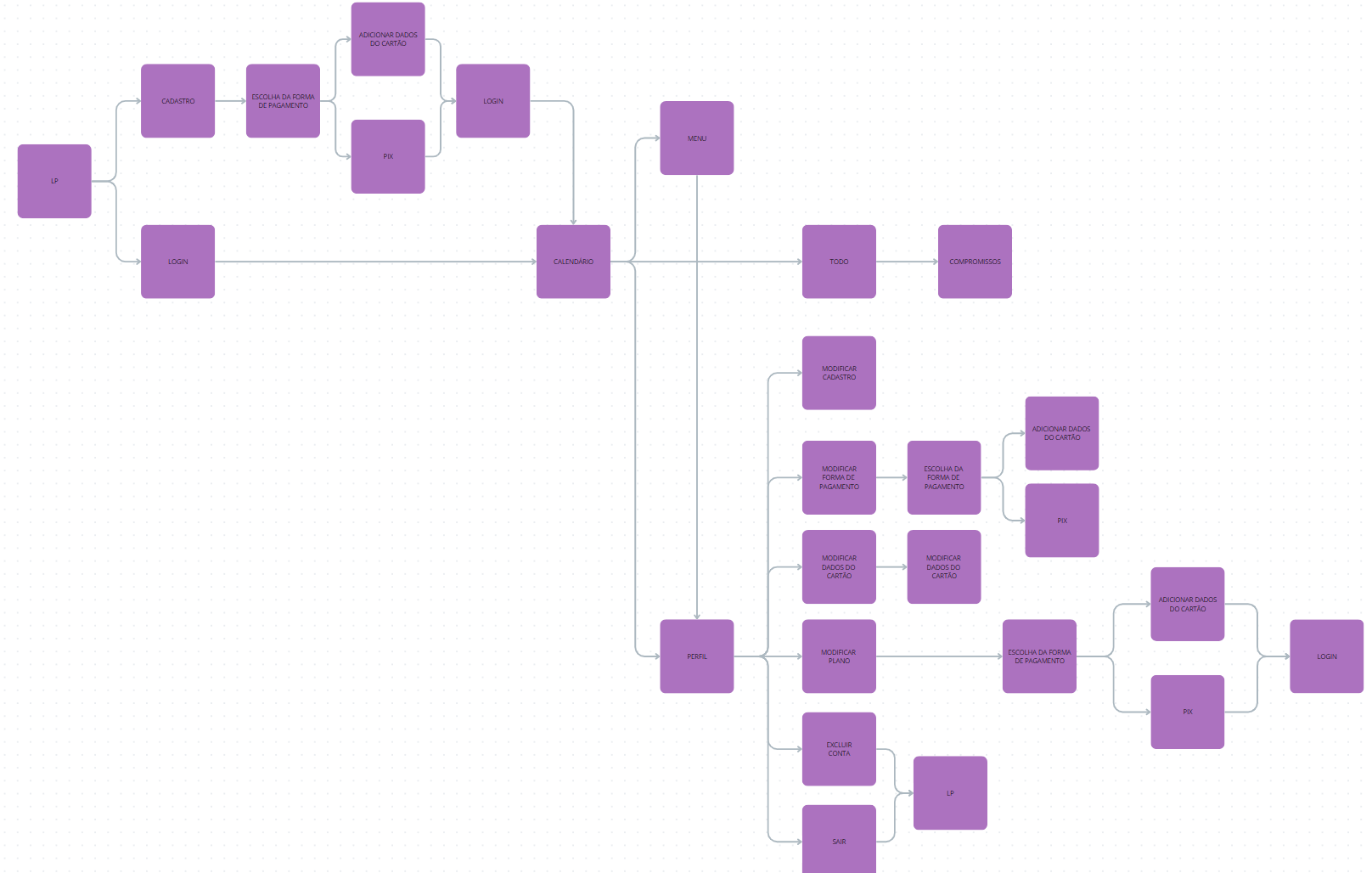
As últimas atualizações feitas na **Sprint 8** foram em relação ao banco de dados correlacionar ao localstorage, ou seja, ao usuário acessar a página para a atualização de seu cadastro os seus dados apareciam já registrados nos campos, tanto seu nome, quanto o seu e-mail, entretanto a senha não, por questões de segurança; além disso, nessa mesma parte de banco, demos funcionalidade ao botão de deletar conta, ou seja, através da relação do delete do banco de dados o usuário poderá realizar a exclusão de seu perfil em nossa aplicação. Outro ponto relevante em nosso projeto se trata da inicialização do nosso TOINTO mobile, criação do wireframe e criação das primeiras telas e rotas.

As últimas atualizações feitas na **Sprint 9** foram relacionadas a criação das telas faltantes em nossa aplicação mobile, suas rotas, juntamente ao desenho em relação a navegação de todo nosso projeto. Também houve a inicialização do nosso CRUD mobile (do cadastro, compromissos e cartão), que inclui o banco de dados e as validações faltantes em nossa aplicação; além disso, realizamos análises em nosso projeto geral com o intuito de dar prioridade a nossos componentes e funções mais importantes e possíveis pendencias, tanto em nosso aplicativo mobile quanto desktop;

1. **Atualizações a fazer**

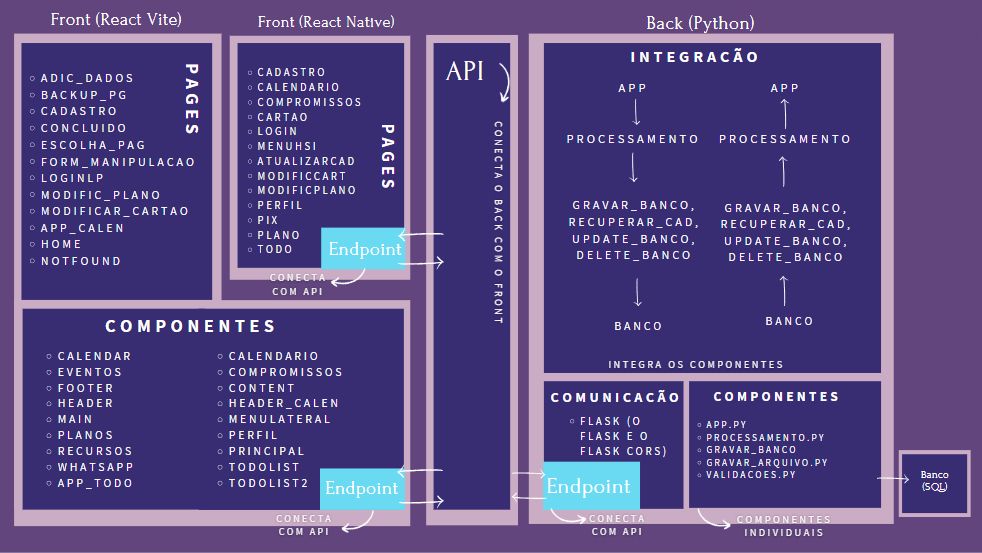
Em nosso projeto foi possível identificar pendências extremamente importantes para melhorar a funcionalidade e a usabilidade da aplicação. Entre elas, trata-se dos compromissos do usuário, que não estão sendo exibidos na tela do calendário, nem na lista de tarefas (TodoList). Devido a essa ausência, os usuários estão impossibilitados de acessar e modificar seus compromissos, sendo prioridades em nossa aplicação.

1. **Fluxo de Navegação**

****

1. **Estrutura de dados**

Desenho do Funcionamento



receber\_dados() : receberá dados Json enviados através de uma solicitação (requisição) POST, e em seguida, retornará um Json com o resultado do processamento.

processar\_dados(dados) : receberá os dados de formulário no Front-end, realizar a validação, e poderá criar, selecionar, alterar e deletar dados salvos no banco de dados

conectar() : irá estabelecer a conexão com o banco de dados MySQL.

Processar\_dados\_cad(dados): receberá os dados de formulário da página de cadastro no Front-end, realizará a validação dos campos e criará um cadastro do usuário no banco de dados.

Processar\_alterar\_cad(dados): receberá os dados de formulário da página de cadastro no Front-end, realizará a validação dos campos e modificará o cadastro do usuário no banco de dados

processar\_dados\_cartao(dados): recebe os dados do formulário enviado pelo Flask, processa esses dados para extração das informações necessárias, valida os campos do cartão e retorna os dados processados, ou uma mensagem de erro caso haja alguma inconsistência nos dados fornecidos.

processar\_dados\_log(dados): recebe os dados de login, como e-mail e senha, e os processa para verificação. Após processar com sucesso, imprime os dados recebidos, executa a validação do e-mail e senha, e retorna uma lista de mensagens de erro, se houver. Caso contrário, executa uma consulta para verificar as informações de login no banco de dados e retorna o resultado dessa consulta.

excluir\_todas\_informações\_usuario(cad\_id): recebe o ID do usuário como entrada e tenta excluir todas as informações desse usuário do banco de dados. Se a exclusão for bem-sucedida, retorna uma mensagem de sucesso indicando que todas as informações foram excluídas. Em caso de erro durante a exclusão, retorna uma mensagem de erro detalhando a exceção ocorrida.

processar\_dados\_compromisso(dados): recebe os dados de um compromisso e os processa para extração das informações necessárias. Após processar esses dados, realiza a validação do título e da descrição do compromisso. Se houver mensagens de erro, retorna uma lista dessas mensagens. Caso contrário, grava os dados do compromisso e retorna uma mensagem de sucesso indicando que os dados foram processados com êxito.

deletar\_cad(cad\_id): recebe o ID do cadastro como entrada e utiliza uma conexão com o banco de dados para excluir o registro correspondente na tabela CADASTRO. Após a execução bem-sucedida do comando SQL, fecha a conexão com o banco de dados e retorna uma mensagem indicando que o usuário foi deletado com sucesso, juntamente com um indicador de erro falso.

gravar\_dados(dados\_gravacao): recebe uma lista contendo os dados de gravação com informações, utiliza uma conexão com o banco de dados para inserir esses dados na tabela cadastro. Se o campo opc for igual a 1, o ID do cadastro é recuperado usando os dados inseridos e é utilizado para inserir uma nova entrada na tabela compras. Após a inserção bem-sucedida dos dados, a conexão com o banco de dados é fechada.

gravar\_dados(dados\_gravacao): insere informações de cadastro na tabela 'cadastro' do banco de dados, utilizando dados como nome de usuário, e-mail e senha. Se opc(opção) for igual a 1, a função também registra uma compra na tabela 'compras' utilizando o ID do cadastro inserido. Após as operações, a conexão com o banco de dados é encerrada para garantir a integridade dos recursos.

gravar\_dados\_compromisso(dados\_gravacao): tem a responsabilidade de inserir informações sobre compromissos na tabela 'compromissos' do banco de dados. Ao receber os dados de gravação, como o ID do cadastro associado aos outros campos, a função executa um comando SQL para inserir esses dados na tabela 'compromissos'. Após a execução bem-sucedida da inserção, a conexão com o banco de dados é fechada, assegurando a integridade dos recursos.

Gravar\_dados\_cartao(dados\_gravacao): Realiza o insert para os dados do cartão sejam inseridos. Após a execução bem-sucedida da inserção, a conexão com o banco de dados é fechada, assegurando a integridade dos recursos.

Verificar\_informacao\_log(email, senha): responsável pela consulta no banco de dados em relação a recuperação das informações de usuário. Se uma correspondência for encontrada, ela retorna, indicando que não há erro e fornece os detalhes do usuário encontrado. Caso contrário, retorna um dicionário indicando que houve um erro, juntamente com uma mensagem informando que o usuário não foi encontrado.

Atualizar\_cad(novos\_dados): altera os dados do usuário na tabela 'cadastro' do banco de dados MySQL. Após a execução bem-sucedida, a conexão com o banco de dados é encerrada e a função retorna um dicionário indicando que não houve erro e que a alteração foi realizada com sucesso.

validar\_nome\_titular(nome\_titular): realiza a validação do campo do nome\_titular em relação a validação do cartão

validar\_cpf(cpf): a função verifica se o cpf digitado é válido ou não, remove todos os caracteres que não sejam dígitos, verifica a quantidade de dígitos. Se houver erros, retorna um dicionário com erros, caso contrário, retorna um dicionário vazio

validar\_num\_cartao(num\_cartao): verifica se o número do cartão é válido, se possui a quantidade de dígitos necessários, se são somente dígitos. Se todas as verificações passarem, a função retorna um dicionário sem erro e uma mensagem vazia, caso contrário, retorna um dicionário indicando um erro, juntamente com uma mensagem explicativa.

validar\_datavenc(datavenc): verifica se não o campo não está vazio, se a data fornecida é válida, anterior a data atual. Se a data fornecida for anterior à data atual, a função retorna um dicionário indicando um erro, com a mensagem 'A data está vencida!'. Caso contrário, retorna um dicionário sem erro e uma mensagem vazia.

validar\_codseg(cod\_seguranca): Verifica se o código de segurança é válido: remove os espaços em branco, verifica se possui apenas dígitos e se possui exatamente três dígitos. a função retorna um dicionário sem erro e uma mensagem vazia. Caso contrário, retorna um dicionário indicando um erro, juntamente com uma mensagem explicativa

validar\_nome(nome): verifica se o nome fornecido é válido: se possui somente letras, se não é menor que 3 caracteres e se não é maior que 100. Se todas as verificações passarem, a função retorna um dicionário sem erro e uma mensagem vazia. Caso contrário, retorna um dicionário indicando um erro, juntamente com uma mensagem explicativa.

validar\_email(vemail): verifica se o e-mail fornecido tem um formato válido. Se estiver no formato correto retorna um dicionário sem erro e uma mensagem vazia. Caso contrário, retorna um dicionário indicando um erro, junto com uma mensagem explicativa.

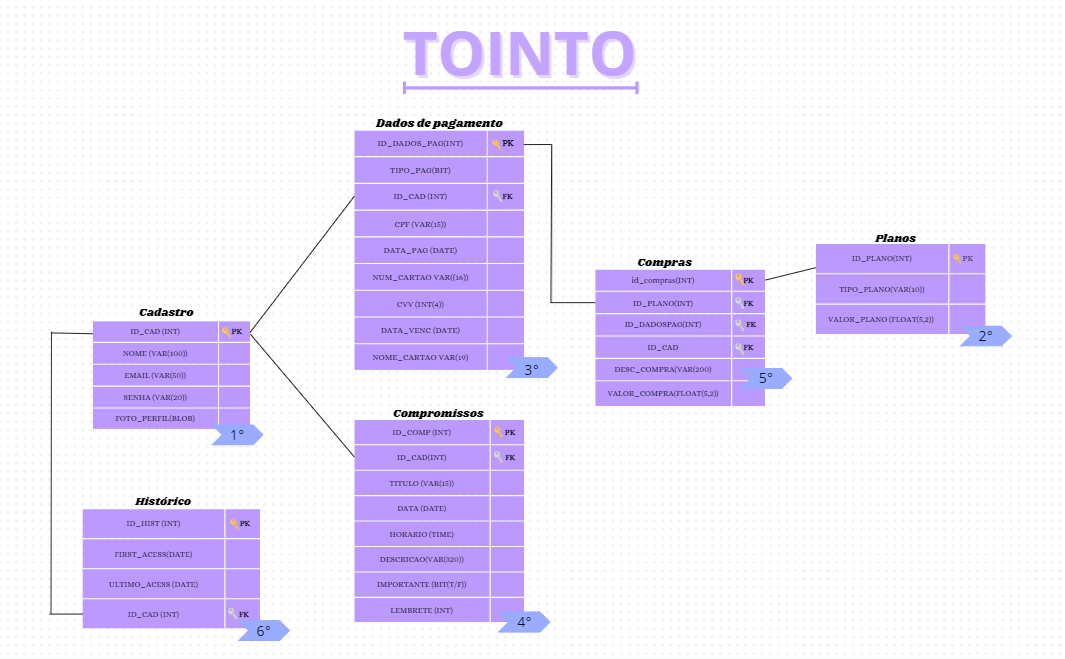
Validar\_senha(vsenha): verifica se a senha fornecida é forte o suficiente, garantindo que tenha entre 8 e 15 caracteres, contenha letras maiúsculas, minúsculas, números e caracteres especiais. Se a senha não cumprir esses requisitos, a função retorna mensagens de erro específicas. Caso contrário, retorna uma mensagem vazia, indicando que a senha é válida.

Confirmar\_senha(senha, confirmsenha): verifica se a senha confirmada é igual à senha original fornecida. Se as senhas não coincidirem, a função retorna um dicionário indicando um erro, juntamente com uma mensagem explicativa. Caso contrário, retorna um dicionário sem erro e uma mensagem vazia.

Validar\_titulo(titulo): verifica se o título do compromisso fornecido é válido, garantindo que tenha entre 3 e 200 caracteres. Se o título estiver fora desses limites, a função retorna uma mensagem de erro. Caso contrário, retorna uma mensagem vazia, indicando que o título é válido

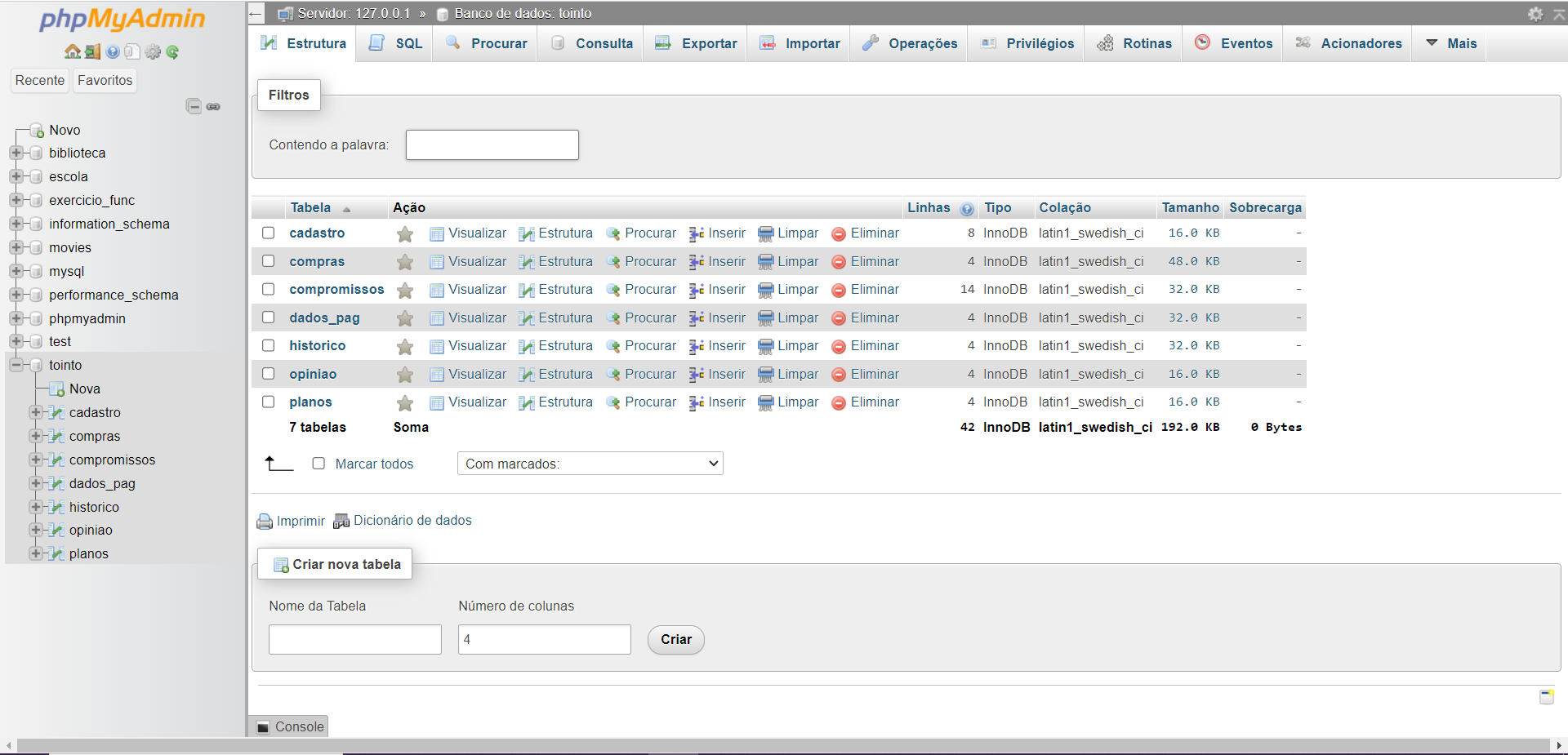
Validar\_desc(desc): verifica se a descrição do compromisso fornecido é válido, assegurando que tenha pelo menos 3 caracteres e não exceda 100 caracteres. Se a descrição não cumprir esses critérios, a função retorna uma mensagem de erro específica. Caso contrário, retorna uma mensagem vazia, indicando que a descrição é válida.

1. **Tabela de relacionamento do TOINTO no Banco de Dados**

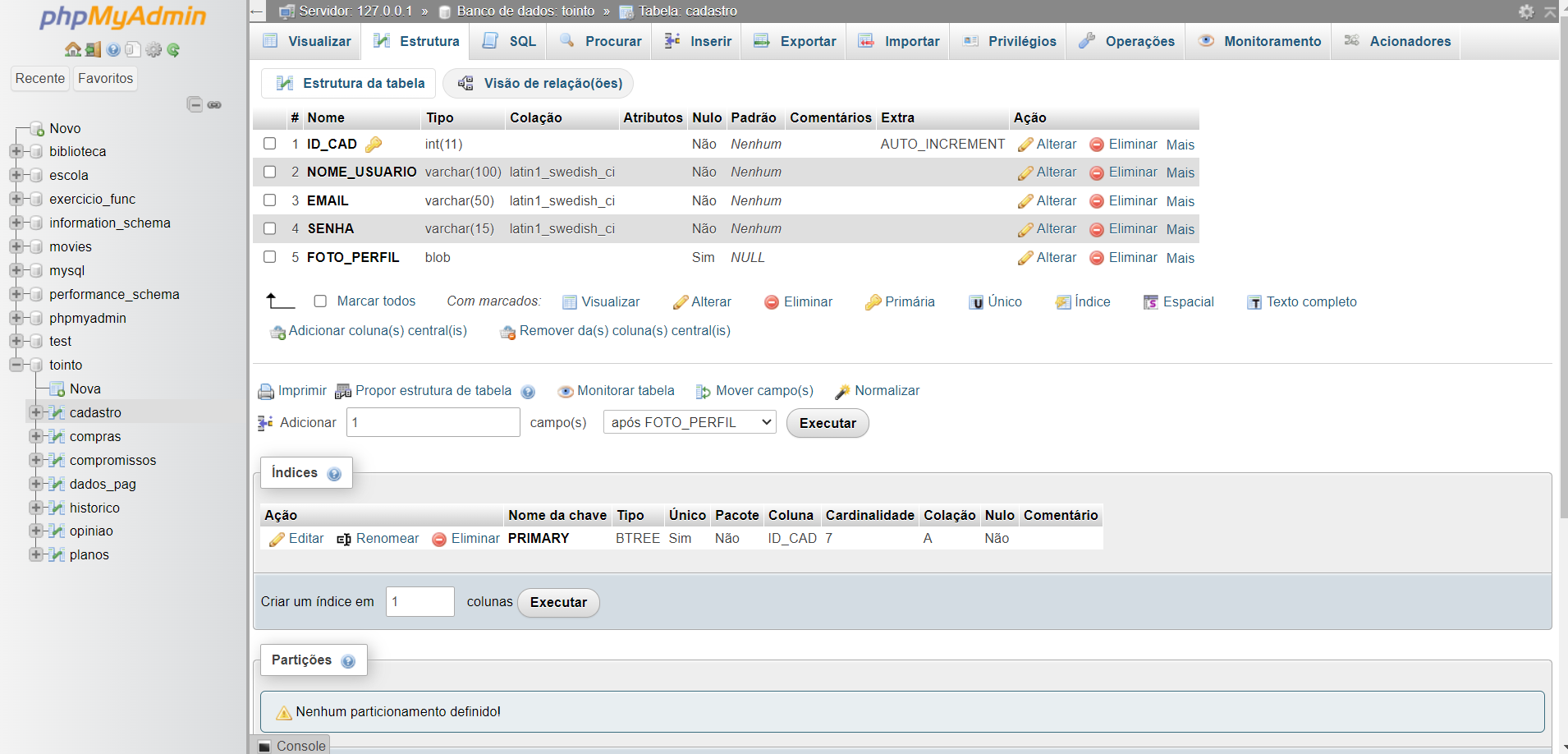


1. **Banco de Dados**

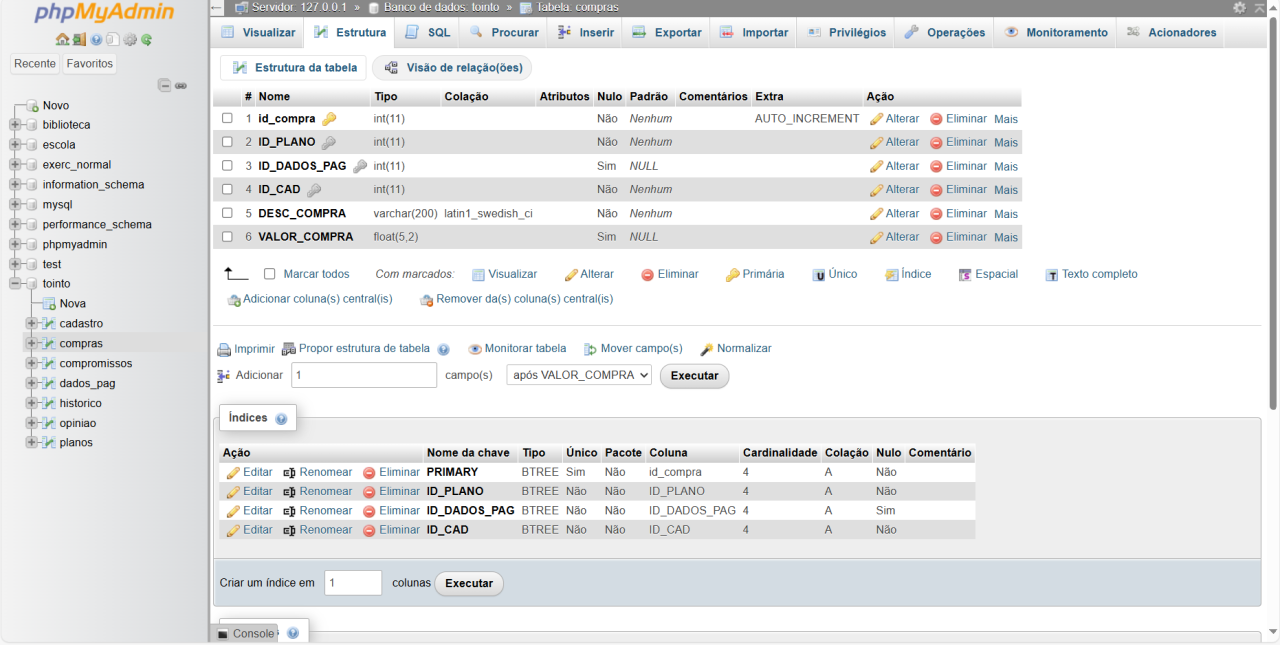
**16.1- Visão geral das tabelas do Banco:**

****

***16.2- Visão da estrutura da tabela “cadastro”:***

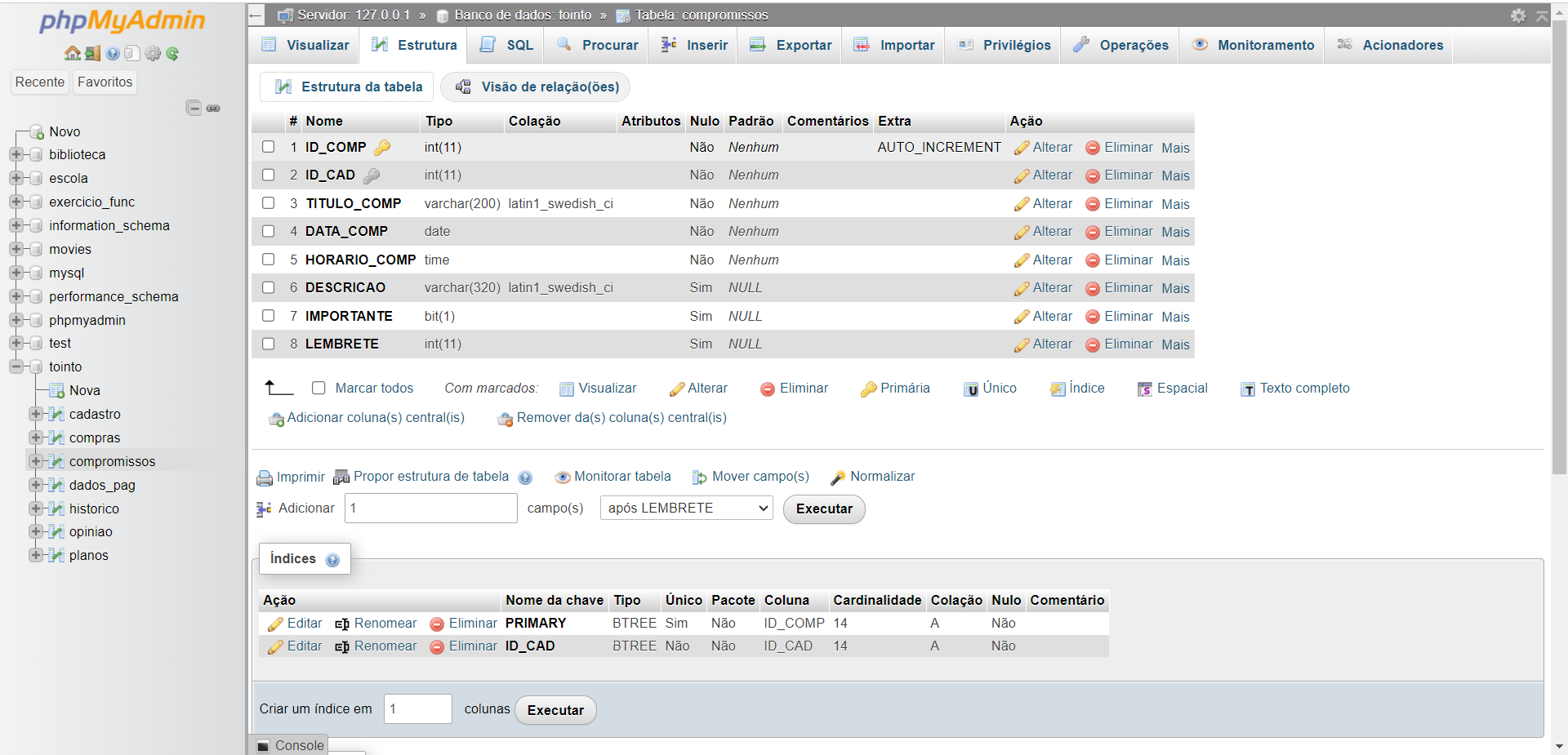
****

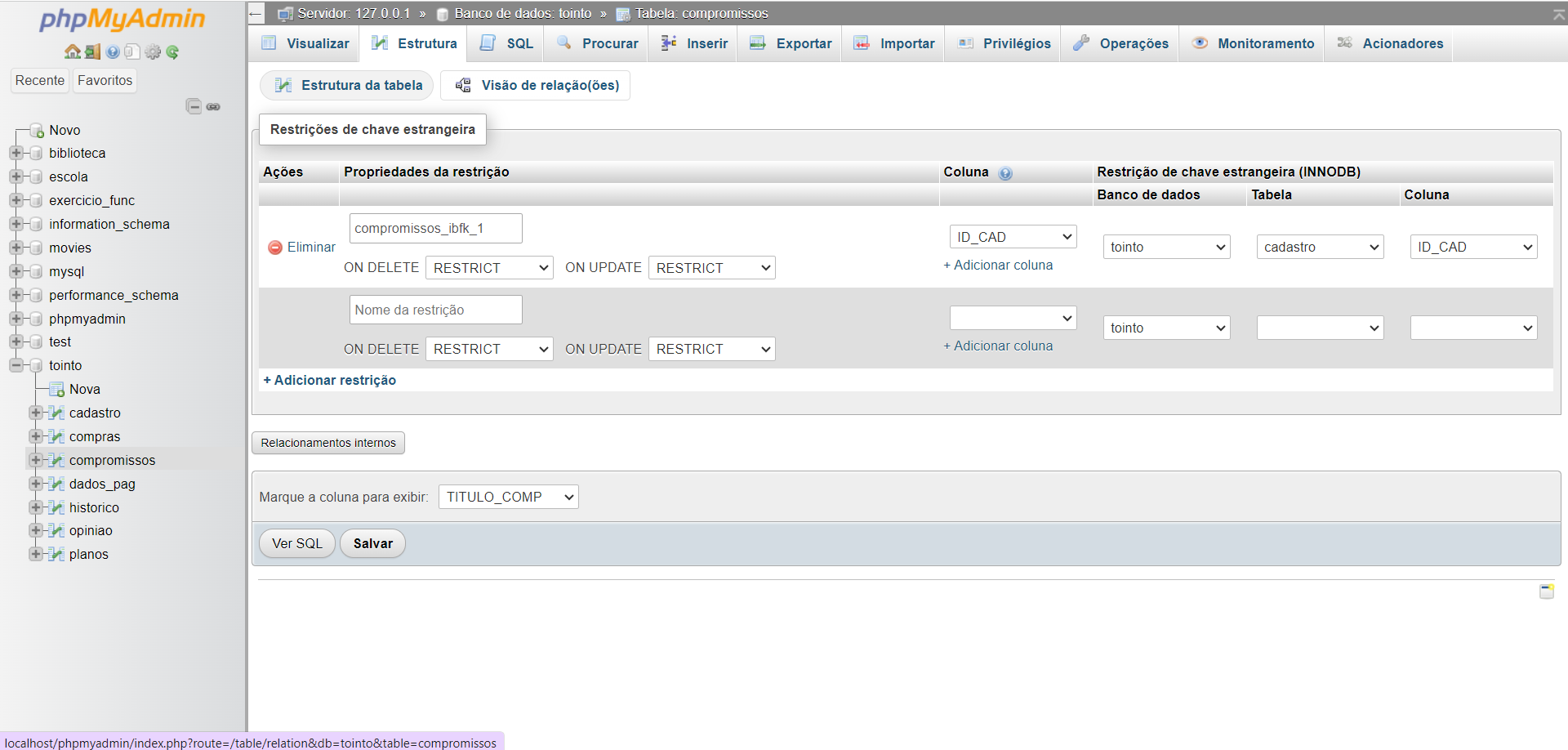
**16.3- Visão da estrutura da tabela “compras”:**

****

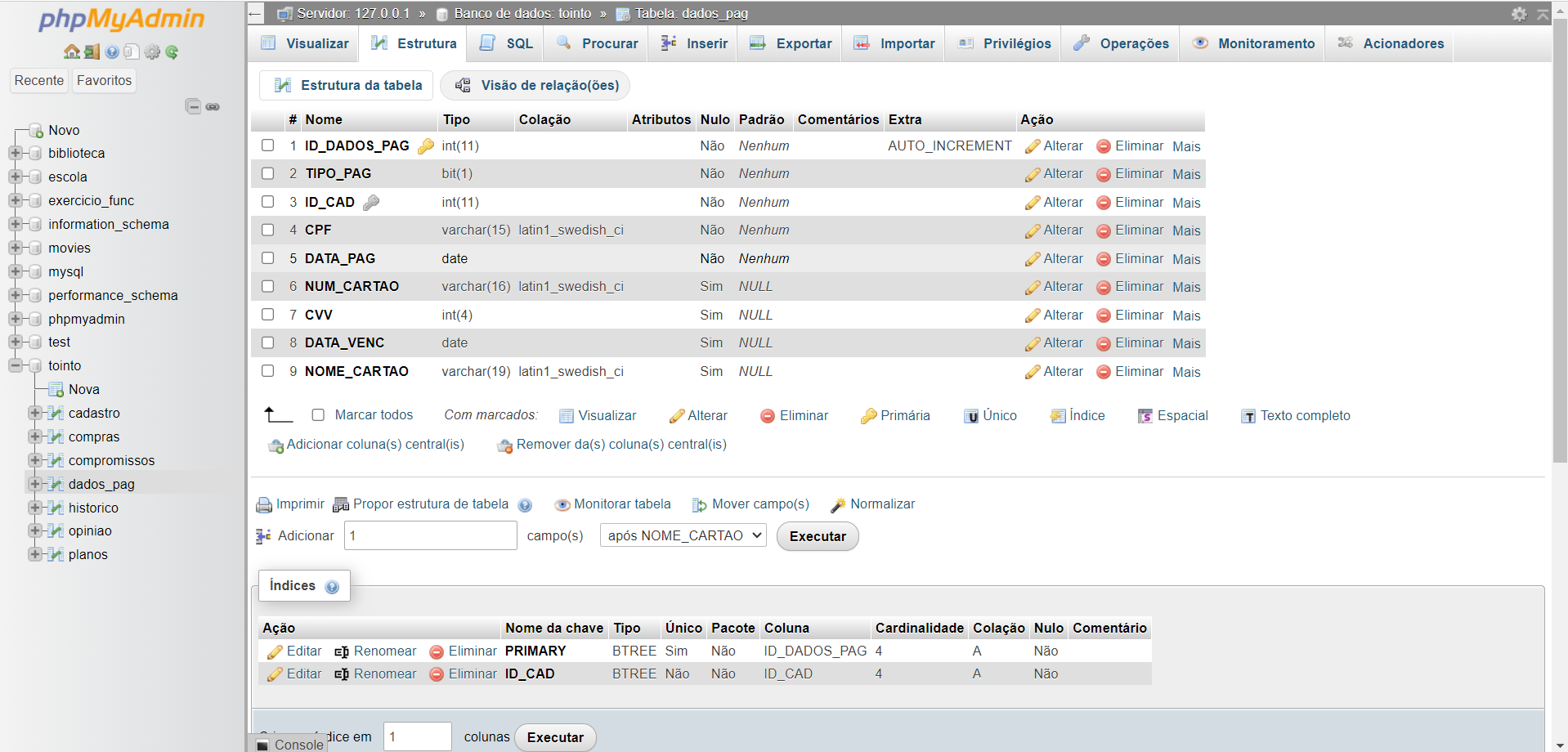
****

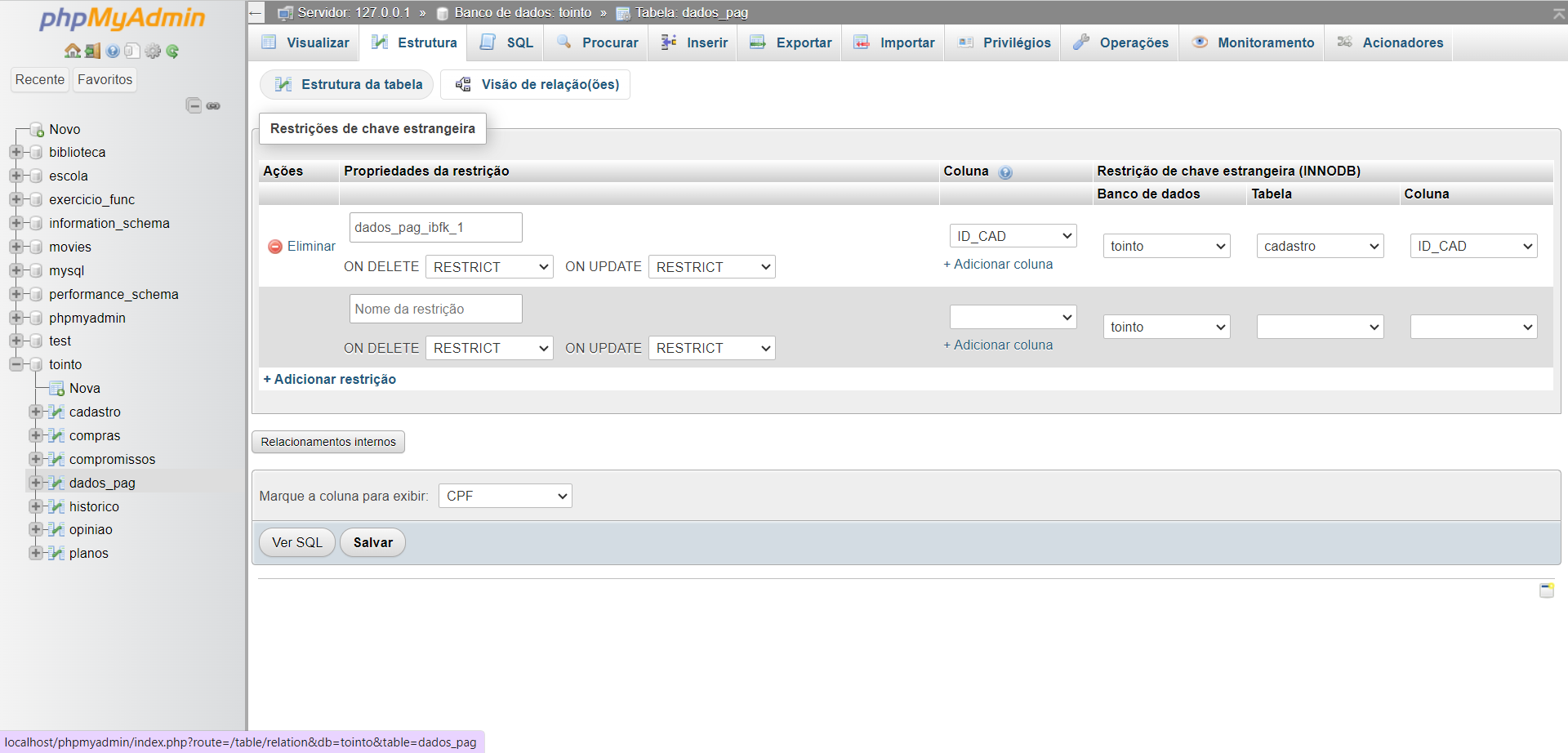
**16.4- Visão da estrutura e relação da tabela “compromissos”:**

****

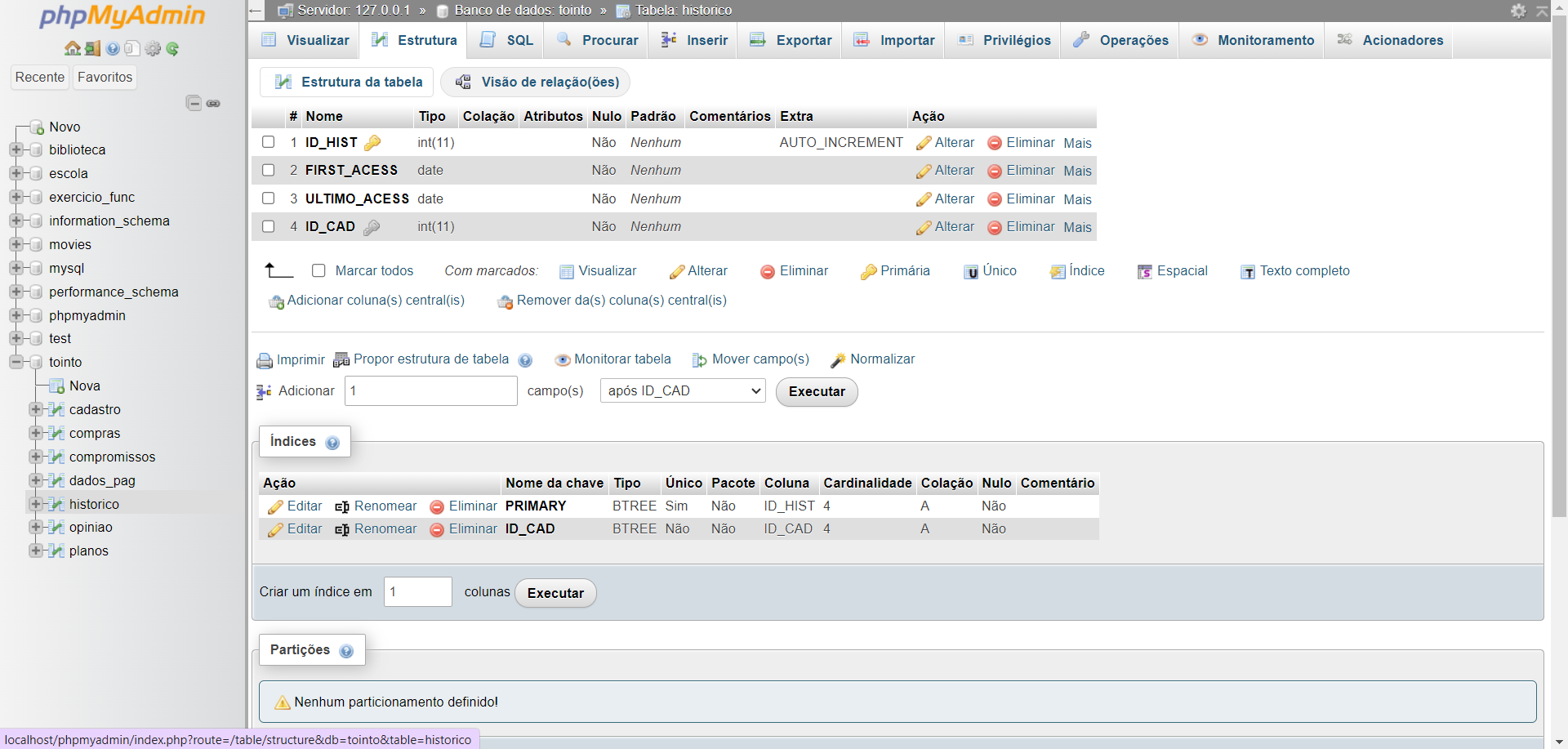
****

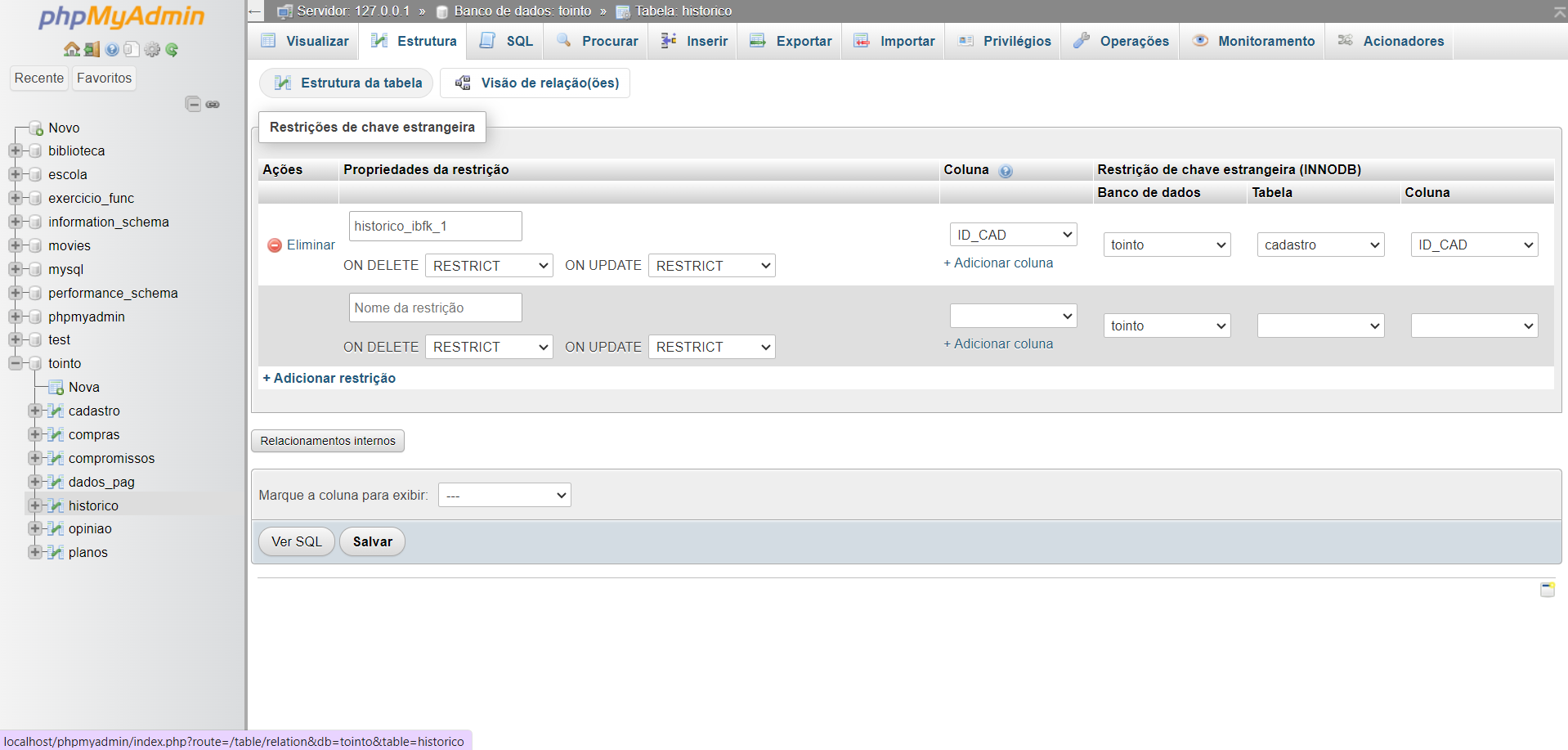
**16.5- Visão da estrutura e relação da tabela “dados\_pag”:**

****

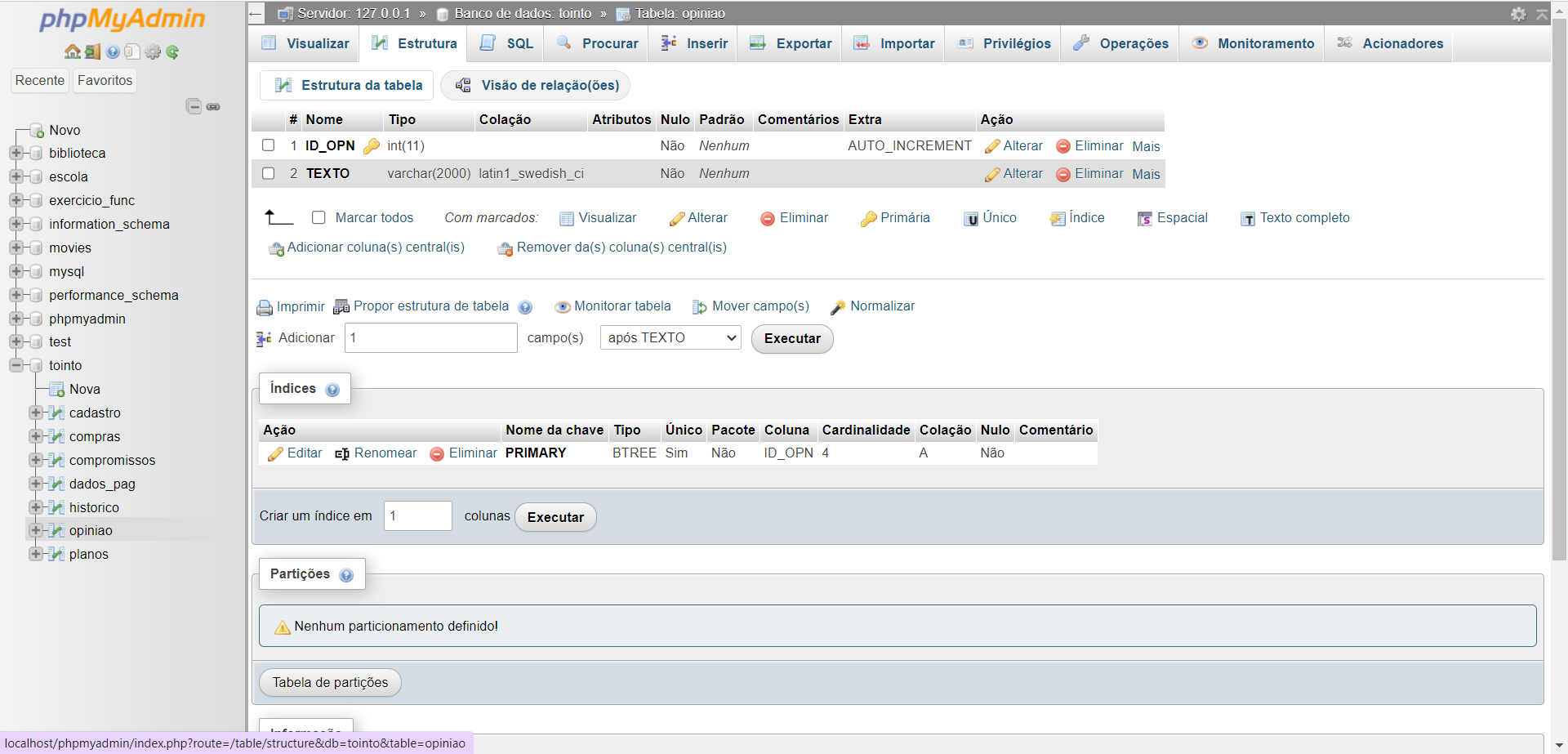
****

**16.6- Visão da estrutura e relação da tabela “historico”:**

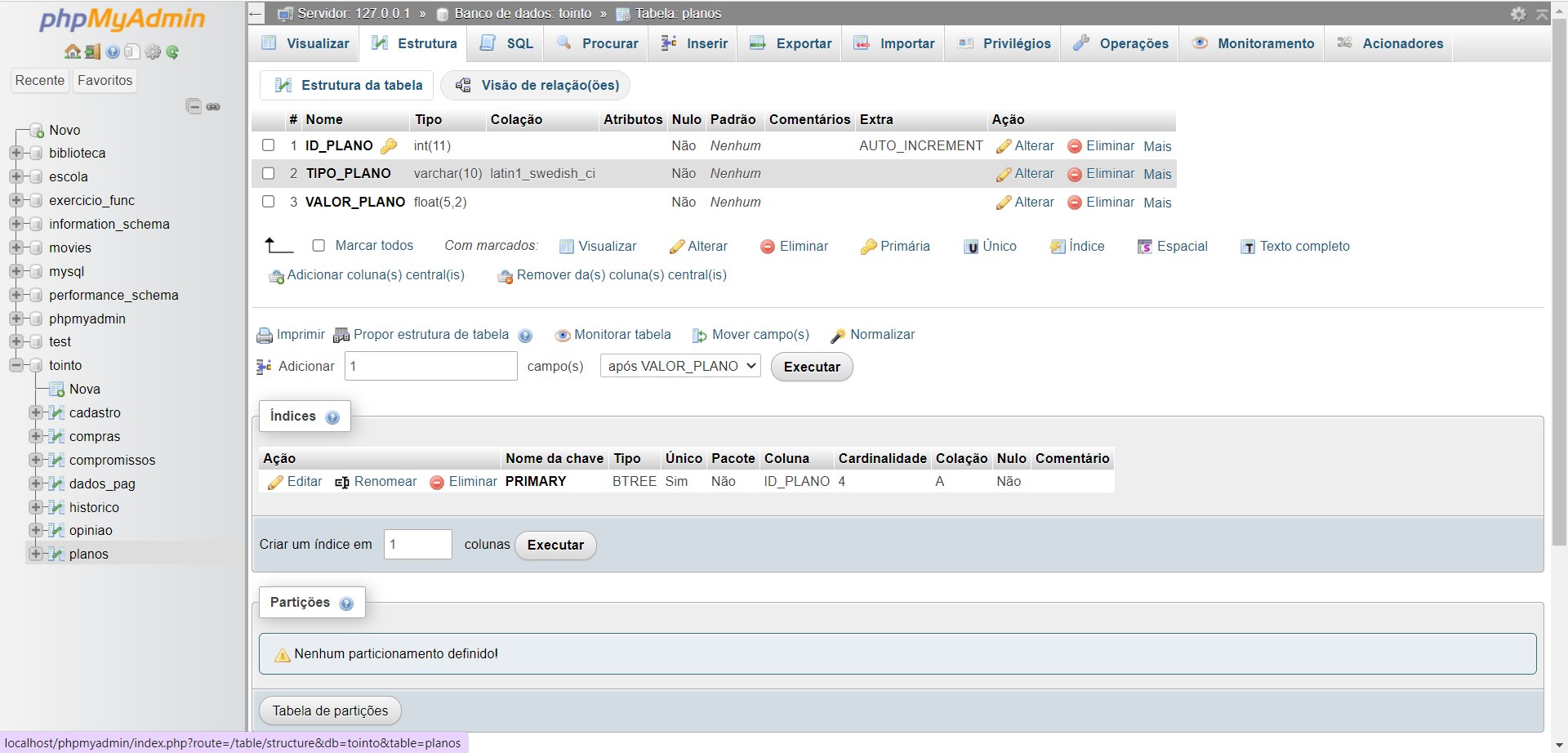
****

****

**16.7- Visão da estrutura da tabela “opiniao”:**

****

**16.8- Visão da estrutura da tabela “planos”:**

****

1. **Documentação do MCU**

**17.1. Lista de Requisitos**

**RFs – Landing Page**

RF1. O sistema deve permitir que o usuário possa realizar o cadastro;

RF2. O sistema deve permitir que o usuário possa logar;

RF3. O sistema deve permitir que o usuário possa ter acesso ao calendário;

RF4. O sistema deve permitir que o usuário possa escolher entre três planos;

RF5. O sistema deve permitir que o usuário possa escolher a forma de pagamento.

RF6. O sistema deve permitir que o usuário possa adicionar as informações do cartão.

RF7. O sistema deve permitir que o usuário possa entrar em contato com o suporte.

**RFs- Página Calendário**

RF1.O sistema deve permitir que o usuário ao clicar no menu, possa ter acesso ao home e as três opções (Hoje, Semana e Importante);

RF2. O sistema deve permitir que o usuário ao clicar no Botão Hoje possa acessar seus compromissos daquele dia, contendo o título, horário, a descrição do evento e os compromissos ligados a importantes;

RF3. O sistema deve permitir que o usuário ao clicar no Botão Semana possa acessar seus compromissos da semana, contendo o dia, título, horário e a descrição dos eventos;

RF4. O sistema deve permitir que o usuário ao clicar no Botão Importantes possa acessar os compromissos que marcou como importante;

RF5. O sistema deve permitir que o usuário ao clicar em home possa voltar para ao calendário.

RF6. O sistema deve permitir que o usuário possa mudar o mês;

RF7. O sistema deve permitir que o usuário ao clicar em algum dia do mês possa visualizar os eventos;

RF8. O sistema deve permitir que o usuário ao clicar em perfil possa ter acesso ao Modificar cadastro, Modificar pagamento e sair do calendário;

RF9. O sistema deve permitir que o usuário ao clicar em Modificar pagamento tenha acesso ao Modificar escolha de pagamento e Modificar dados do cartão;

RF10. O sistema deve permitir que o usuário ao clicar em Modificar escolha de pagamento, possa escolher entre pix ou cartão;

RF11. O sistema deve permitir que o usuário ao clicar em cartão possa adicionar os dados do cartão;

RF12. O sistema deve permitir que o usuário ao clicar em Modificar dados do cartão possa atualizar os dados do cartão;

**RFs – Página do To Do List**

RF1. O sistema deve permitir que o usuário, na página de TO DO LIST, possa editar, adicionar ou excluir compromissos;

RF2. O sistema deve permitir que o usuário marque como concluído os compromissos da página do TO DO LIST.

RF3. O sistema deve permitir que o usuário possa criar os seus compromissos;

**RFs – Página do Compromissos**

RF1. O sistema deve permitir que o usuário preencha os campos editáveis (horário, título da tarefa, descrição, importante e lembrete)

RF2. O sistema deve permitir que o usuário salve os compromissos.

**17.2. Atores**

* **Usuário:** indivíduo que utilizará o site, desfrutando de todas funcionalidades disponíveis;

1. **Usuário Gratuito;**
2. **Usuário Mensal;**
3. **Usuário Anual.**

* **Sistema bancário:** meio de pagamento para transações bancárias.
* **Sistema:** realizará algumasações sozinho, nele mesmo.

**17.3. Fluxo principal – Landing Page:**

1. O usuário escolhe um plano.
2. O usuário solicita o cadastro.
3. O sistema verifica se os dados estão corretos;
4. Se estiver incorreto o sistema mostra uma mensagem de erro;
5. Se estiver correto o sistema armazena os dados do usuário;
6. Caso o usuário escolha um plano pago;
7. O sistema apresenta o cadastro;
8. O sistema apresenta duas escolhas de pagamento para o usuário.
9. O usuário escolhe uma das formas de pagamento;
10. Se o usuário escolher pix, o sistema mostra o QRCODE
11. Se o usuário escolher cartão, o sistema apresenta a página de adicionar dados do cartão;
12. O usuário realiza o pagamento;
13. O banco valida o pagamento;
14. O sistema é notificado;
15. O usuário solicita o login;
16. O sistema analisa os dados;
17. Caso forem incorretos, o sistema mostra uma mensagem de erro;
18. Caso forem corretos, o sistema permite que o usuário entre no calendário;
19. Caso usuário solicite o suporte;
20. O sistema encaminha o usuário para o Whatsapp;
21. Caso o usuário envie uma mensagem;
22. O sistema armazena;
23. Caso o usuário acesse uma das redes sociais;
24. O sistema encaminha para a conta do TOINTO

**17.4. Fluxo principal – Página calendário:**

1. O usuário entra no calendário;
2. Caso a escolha tenha sido grátis, o sistema disponibiliza apenas 5 compromissos ao dia para o usuário e não disponibiliza os recursos de Hoje, Semana, Importantes que estão no menu.
3. Caso a escolha tenha sido plano mensal, o sistema disponibiliza 10 compromissos ao dia para o usuário e acesso ao item Hoje e Semana do menu.
4. Caso a escolha tenha sido plano anual, o sistema disponibiliza 25 compromissos ao dia e acesso a todos os itens do menu.
5. O usuário clica no botão Hoje.
6. O sistema abre os compromissos do usuário daquele dia, contendo a Data, Nome, Horário, descrição dos eventos e os marcados como importantes.
7. O usuário clica no botão Semana.
8. O sistema abre os compromissos do usuário daquela semana, que contém a Data, Nome, Horário, a descrição dos eventos e os marcados como importantes.
9. O usuário clica no botão Importantes.
10. O sistema abre os compromissos que o usuário marcou como importante.
11. O usuário muda o mês.
12. O sistema leva o usuário para o próximo mês.
13. O usuário muda o ano.
14. O sistema leva o usuário ao próximo ano.
15. O usuário clica em um dia do mês.
16. O sistema mostra os eventos daquele dia, caso tenha, e possibilita que ele possa editá-lo pelo botão Lápis, excluí-lo pelo botão Lixo ou adicionar tarefas pelo botão mais.
17. Caso o usuário não tenha adicionado nenhum evento naquele dia, o sistema mostra o TO DO LIST sem nenhum evento e disponibiliza um botão para que ele possa adicionar eventos.
18. O usuário clica no botão de adicionar compromissos.
19. O sistema leva o usuário para a página de compromissos, no qual é mostrado os campos para ele preencher, com a data fixa.
20. Caso o usuário opte por ser lembrado algum tempo antes, ele seleciona uma opção de lembrete.
21. O sistema salva a opção escolhida para notificá-lo no tempo determinado.
22. O usuário clica em salvar.
23. O sistema salva o compromisso no dia correspondente;
24. O usuário clica no perfil;
25. O sistema apresenta os botões de Modificar cadastro, Modificar pagamento e sair
26. Caso o usuário clique em Modificar cadastro, o sistema apresenta o cadastro com as informações salvas anteriormente;
27. O usuário muda as informações e clica em salvar;
28. O sistema armazena as informações atualizadas;
29. Caso o usuário clique em Modificar pagamento, o sistema apresenta os subtópicos Modificar escolha do pagamento e Modificar dados do cartão;
30. Caso o usuário clique em Modificar escolha do pagamento, o sistema apresenta a página de escolha de pagamento com as informações salvas anteriormente;
31. Se o usuário clicar em pix, o sistema apresenta a página com o QRCODE;
32. Se o usuário clica em cartão, o sistema apresenta a página de adicionar dados do cartão;
33. O usuário adiciona as informações necessárias e clica em salvar;
34. O usuário realiza o pagamento;
35. O banco valida o pagamento;
36. O sistema é notificado;
37. Caso o usuário clique em Modificar dados do cartão, o sistema apresenta a página com os dados gravados anteriormente;
38. O usuário modifica os dados e clica em salvar;
39. O sistema armazena os dados atualizados;