# DOCUMENTAÇÃO DA APLICAÇÃO TASK MANAGER (APP DE AGENDAS E TAREFAS)

## VISÃO GERAL

Introdução: A aplicação Task Manager é um sistema de gerenciamento de tarefas que permite criar, editar, excluir e imprimir relatórios. O objetivo principal da aplicação é facilitar a organização e o acompanhamento das tarefas diárias. Com o Task Manager, os usuários podem manter um registro das tarefas pendentes, atribuir responsabilidades, definir prazos e monitorar o status de conclusão.

Tecnologias Utilizadas: A construção da aplicação Task Manager envolveu o uso das seguintes tecnologias:

* Linguagens de Programação: HTML, CSS, PHP 8 e JavaScript.
* Frameworks e Bibliotecas: Bootstrap 5.3.0-alpha3, jQuery 3.7 e DataTables 1.10.25.
* Banco de Dados: MySQL 8.0.30.
* Outras Ferramentas: Select2 para seleção avançada de elementos, SweetAlert2 para exibição de mensagens de confirmação e pop-ups, e bibliotecas adicionais para suporte de exportação de dados em PDF, Excel e impressão.

## CONFIGURAÇÃO E INSTALAÇÃO

Requisitos do Sistema**:** Para executar a aplicação Task Manager, é necessário atender aos seguintes requisitos mínimos do sistema:

* Servidor Apache
* PHP 8
* Servidor MySQL 8
* Se quiser facilitar, instale o xampp ou o laragon que já vem com os 3 serviços instalados e pré configurados. No caso do xampp, a aplicação deverá ficar dentro de htdocs e no caso do laragon no www.

Instalação: Siga os passos abaixo para instalar e configurar a aplicação no ambiente desejado:

1. Clone o repositório do Task Manager no GitHub. Você pode fazer isso executando o seguinte comando no terminal:

git clone https://github.com/vitorpedroso283/TaskManagerApp

1. Certifique-se de que o servidor Apache, o PHP 8 e o servidor MySQL estejam instalados e devidamente configurados em sua máquina.
2. Inicie o servidor Apache e o servidor MySQL.
3. Verifique se o servidor de banco de dados MySQL está sendo executado na porta 3306 e o usuário padrão é "root" sem senha. Caso contrário, atualize as informações de conexão no arquivo de configuração apropriado.
4. Importe o arquivo de dump SQL. Na raiz do projeto Task Manager, você encontrará um arquivo com o nome "dump\_task\_manager.sql". Crie um schema chamado task\_manager e selecione o database. Utilize a ferramenta de gerenciamento de banco de dados de sua preferência para importar esse arquivo e criar o esquema do banco de dados necessário para a aplicação. Sugestão: DBeaver ou WorkBench. Ambas ferramentas possuem a configuração para import de dump.
5. Configure o servidor web (Apache) para apontar para a pasta do projeto Task Manager.
6. Abra o navegador da web e acesse a aplicação digitando a URL correspondente ao diretório em que você configurou o projeto.
7. O usuário padrão "master" é "[admin@admin.com](mailto:admin@admin.com)" e a senha é "Admin@1234". Você pode usar essas credenciais para fazer login na aplicação como administrador. Lembre-se de alterar essas informações e criar novos usuários com senhas seguras após a instalação, para garantir a segurança da sua aplicação.
8. Novos usuários cadastrados virão com a permissão apenas de leitura, sendo que poderão ser trocadas posteriormente por um usuário master.

Certifique-se de ajustar as configurações e informações de conexão de acordo com o seu ambiente específico.

### Passo a passo de uso da aplicação:

1. Tela de Login:
   * Nesta tela, o usuário deve inserir seu nome de usuário (email) e senha para fazer o login.
   * Caso não possua uma conta, o usuário pode clicar no link "Cadastre-se" para criar uma nova conta.
2. Tela de Cadastro:
   * Na tela de cadastro, o usuário deve preencher as seguintes informações:
     + Nome completo
     + Telefone
     + Email único (será utilizado como nome de usuário)
     + Senha seguindo os padrões de uma senha forte.
3. Permissões de usuário:
   * Ao criar uma nova conta, o usuário terá apenas a permissão de leitura (visualização) no sistema.
   * As demais permissões serão atribuídas pelo administrador.
4. Tela inicial (Home) - Tarefas Pendentes:
   * Após fazer o login, o usuário será redirecionado para a tela inicial, que exibirá as tarefas pendentes.
   * Nessa tela, o usuário poderá visualizar as tarefas pendentes e alterar o status de uma tarefa para "Concluído".
5. Permissões das tarefas:
   * Existem diferentes permissões relacionadas às tarefas:
     + Ler(VISUALIZAR/READ/R): permite visualizar e trocar o status de uma tarefa para "Concluído".
     + Atualizar(EDITAR/UPDATE/U): permite editar as informações de uma tarefa.
     + Excluir(DELETAR/DELETE/D): permite excluir uma tarefa.
     + Imprimir(IMPRIMIR/PRINT/P): permite gerar relatórios em formato Excel e PDF.
     + Criar(INSERIR/CREATE/C): permite criar novas tarefas.
     + Admin(ADMIN/A): permite criar novos usuários, alterar permissões dos usuários e voltar uma tarefa como "Pendente".
6. Criar nova tarefa:
   * Na tela inicial (Home), há a opção de criar uma nova tarefa.
   * Ao selecionar essa opção, um modal será aberto, onde o usuário poderá inserir o nome da tarefa e uma descrição.
   * Ao salvar, a tarefa será adicionada na tabela sem a necessidade de recarregar a página.
7. Tabela de tarefas:
   * Na tabela de tarefas, serão exibidas as seguintes colunas:
     + Nome da tarefa
     + Descrição
     + Data de criação
     + Data de conclusão
     + Status (verde para concluído e amarelo para pendente)
     + Coluna "Concluído" com uma caixa de seleção para marcar a tarefa como concluída.
     + Coluna "Ações" com botões de "Excluir" e "Editar".
8. Cabeçalho (nav, header):
   * No cabeçalho da página, existem as seguintes opções:
     + "Todas as tarefas": redireciona para uma tela semelhante à de tarefas pendentes, mas exibe todas as tarefas, incluindo as concluídas.
     + "Controle de Acesso": acesso restrito aos usuários com permissão de admin. Nessa tela, é possível editar informações do usuário, como senha, telefone e nome, além de gerenciar as permissões do usuário, adicionando ou removendo.
     + Botão "Sair": encerra a sessão e faz o logout do usuário.

Observação:

* As informações de permissões são armazenadas na sessão quando o usuário faz o login, permitindo recuperá-las e fazer as verificações necessárias durante o uso da aplicação.

## ARQUITETURA DA APLICAÇÃO

A aplicação Task Manager segue a arquitetura MVC (Model-View-Controller), que é um padrão de design de software amplamente utilizado para separar as preocupações relacionadas à lógica de negócio, à interface do usuário e ao gerenciamento de dados. Essa arquitetura proporciona uma organização eficiente e uma divisão clara das responsabilidades, facilitando o desenvolvimento, a manutenção e o teste da aplicação.

Estrutura de Pastas

A estrutura de pastas do projeto Task Manager é organizada da seguinte forma:

- app/ --> Pasta principal da aplicação

- config/ --> Arquivos de configuração

- database.php --> Configurações do banco de dados

- authenticator.php --> Configurações de autenticação

- routes.php --> Configurações de rotas

- controllers/ --> Controladores da aplicação

- TaskManagerController.php --> Controlador para gerenciamento de tarefas

- LoginController.php --> Controlador para autenticação e login

- AccessControlController.php --> Controlador para controle de acesso

- models/ --> Modelos da aplicação

- TaskManagerModel.php --> Modelo para gerenciamento de tarefas

- LoginModel.php --> Modelo para autenticação e login

- AccessControlModel.php --> Modelo para controle de acesso

- routers/ --> Roteadores da aplicação

- accessControlRouter.php --> Roteador para controle de acesso

- loginRouter.php --> Roteador para autenticação e login

- taskManagerRouter.php --> Roteador para gerenciamento de tarefas

- views/ --> Visualizações da aplicação

- access-control/ --> Visualizações relacionadas ao controle de acesso

- index.php --> Página inicial do controle de acesso

- modals/ --> Modais do controle de acesso

- edit\_permission\_modal.php --> Modal para edição de permissões

- edit\_user\_modal.php --> Modal para edição de usuários

- all-tasks/ --> Visualizações relacionadas a todas as tarefas

- index.php --> Página principal de todas as tarefas

- errors/ --> Visualizações de erros

- error\_403.php --> Página de erro 403 (Acesso negado)

- error\_404.php --> Página de erro 404 (Página não encontrada)

- error\_500.php --> Página de erro 500 (Erro interno do servidor)

- home/ --> Visualizações relacionadas à página inicial

- index.php --> Página inicial do Task Manager

- modals/ --> Modais da página inicial

- create\_task\_modal.php --> Modal para criação de tarefas

- edit\_task\_modal.php --> Modal para edição de tarefas

- includes/ --> Arquivos de inclusão comum

- header.php --> Cabeçalho comum a todas as páginas

- libs.php --> Bibliotecas comuns

- login/ --> Visualizações relacionadas ao login

- index.php --> Página de login

- modals/ --> Modais do login

- register\_user\_modal.php --> Modal para registro de usuários

- public/ --> Arquivos públicos

- css/ --> Arquivos CSS

- login.css --> Estilos para a página de login

- img/ --> Imagens utilizadas na aplicação

- logo.png --> Logo da aplicação

- js/ --> Arquivos JavaScript

- accessControl/ --> JavaScript relacionado ao controle de acesso

- accessControl.js --> Lógica do controle de acesso

- allTasks/ --> JavaScript relacionado a todas as tarefas

- allTasks.js --> Lógica de todas as tarefas

- home/ --> JavaScript relacionado à página inicial

- home.js --> Lógica da página inicial

- login/ --> JavaScript relacionado ao login

- login.js --> Lógica do login

- utils/ --> JavaScript utilitário

- inputMask.js --> Funções de máscara de entrada

- sweetAlertToast.js --> Funções de exibição de alertas

- vendor/ --> Bibliotecas e dependências do projeto

- jquery/ --> Biblioteca jQuery

- jquery.min.js --> Versão minificada do jQuery 3.7.0

- bootstrap/ --> Framework Bootstrap

- bootstrap.bundle.min.js --> Versão minificada do Bootstrap 5.3.0-alpha3

## FLUXO DE DADOS

### O fluxo de dados na aplicação Task Manager ocorre da seguinte maneira:

1. O usuário interage com a interface do usuário, realizando uma ação, como criar, editar ou excluir uma tarefa.
2. O arquivo JavaScript correspondente à ação desejada é acionado. Por exemplo, ao clicar no botão de criação de uma tarefa, o arquivo home.js é executado.
3. O arquivo JavaScript, usando a biblioteca jQuery, realiza uma solicitação AJAX para o arquivo do roteador correspondente à ação. Por exemplo, no caso de criar uma tarefa, a solicitação seria enviada para o arquivo taskManagerRouter.php.
4. O roteador, responsável por direcionar as solicitações, analisa a solicitação recebida e identifica a ação desejada.
5. O roteador instancia a classe do controlador apropriado para lidar com a ação. Por exemplo, no caso de criar uma tarefa, o controlador TaskManagerController é instanciado.
6. O controlador recebe a solicitação e executa a lógica de negócio necessária. Ele pode interagir com a classe do modelo correspondente para buscar ou modificar os dados no banco de dados. Por exemplo, o controlador TaskManagerController pode chamar o método createTask() do modelo TaskManagerModel.
7. O modelo se comunica com o banco de dados utilizando as configurações especificadas no arquivo database.php para buscar ou modificar os dados solicitados. Por exemplo, o modelo TaskManagerModel pode executar uma consulta SQL para inserir uma nova tarefa no banco de dados.
8. O modelo retorna os dados solicitados para o controlador.
9. O controlador, com base nos dados retornados pelo modelo, decide qual visualização deve ser renderizada. Por exemplo, após criar uma tarefa com sucesso, o controlador pode decidir renderizar a visualização index.php da pasta home.
10. A visualização é renderizada com os dados fornecidos pelo controlador e retornada como resposta para a solicitação AJAX.
11. O arquivo JavaScript, ao receber a resposta AJAX, processa os dados retornados e atualiza a interface do usuário, refletindo as alterações realizadas.

Esse fluxo de dados permite uma interação fluida entre a interface do usuário, o JavaScript, os controladores e modelos da aplicação, garantindo a correta manipulação e atualização dos dados no banco de dados e a exibição dos resultados na interface do usuário.

## DESCRIÇÃO DE API’S E FUNCIONALIDADES DO SISTEMA

### API de Controle de Acesso (/app/routers/accessControlRouter.php):

Essa é uma API PHP de controle de acesso que permite realizar operações relacionadas a usuários e permissões. A classe AccessControlController é responsável por receber as requisições e chamar os métodos apropriados da classe AccessControlModel. Ela verifica as permissões antes de executar determinadas operações.

Alguns dos principais métodos da classe AccessControlController são:

* getUsers(): Retorna todas as informações dos usuários, juntamente com suas permissões.
* getUserById(): Retorna as informações de um usuário com base no ID fornecido.
* check(): Verifica se um usuário possui permissão com base no ID fornecido.
* editUser(): Edita as informações de um usuário com base nos dados fornecidos.
* getUserPermissions(): Retorna as permissões do usuário atualmente logado.
* getPermissionsByUser(): Retorna as permissões de um usuário com base no ID fornecido.
* editUserPermissions(): Edita as permissões de um usuário com base nos dados fornecidos.

A classe AccessControlModel realiza as operações de consulta, atualização e exclusão no banco de dados relacionadas aos usuários e permissões. Ela utiliza a classe Connection para estabelecer a conexão com o banco de dados.

Essa API permite visualizar, criar, editar e excluir usuários, bem como gerenciar suas permissões. Também é possível verificar as permissões de um usuário e editar suas permissões.

### API gerenciador de tarefas(/app/routers/taskManagerRouter.php):

Essa é uma API PHP que implementa um sistema de gerenciamento de tarefas. A classe TaskManagerModel é responsável por interagir com o banco de dados e possui métodos para obter tarefas, criar tarefas, editar tarefas, excluir tarefas, visualizar tarefas e atualizar o status de uma tarefa.

A classe TaskManagerController é responsável por receber as requisições e chamar os métodos apropriados da classe TaskManagerModel. Ela também realiza verificação de permissões antes de executar determinadas operações.

Os principais métodos da classe TaskManagerController são:

* getTasks(): Retorna todas as tarefas ou apenas as tarefas pendentes, dependendo do parâmetro callGetTasks passado na requisição.
* createTask(): Cria uma nova tarefa com base nos dados fornecidos na requisição.
* editTask(): Edita uma tarefa existente com base nos dados fornecidos na requisição.
* deleteTask(): Exclui uma tarefa com base no ID fornecido na requisição.
* viewTask(): Retorna os detalhes de uma tarefa com base no ID fornecido na requisição.
* updateTaskStatus(): Atualiza o status de uma tarefa com base no ID e no status fornecidos na requisição.

A classe TaskManagerModel realiza as operações de consulta, inserção, atualização e exclusão no banco de dados. Ela utiliza a classe Connection para estabelecer a conexão com o banco de dados, e possui métodos para obter todas as tarefas, obter tarefas pendentes, criar tarefa, editar tarefa, excluir tarefa, visualizar tarefa e atualizar o status da tarefa.

Ambas as classes dependem do arquivo routes.php para importar as configurações de rota e localização dos arquivos necessários.

Essa API implementa um sistema simples de gerenciamento de tarefas, onde é possível realizar operações como criar, editar, excluir, visualizar e atualizar o status das tarefas.

### API de Login (loginRouter.php):

A API de Login mostrada acima é responsável por lidar com as funcionalidades de login e registro de usuários. Ela segue uma arquitetura MVC (Model-View-Controller) para separar as responsabilidades e facilitar a manutenção do código.

A classe LoginController é responsável por receber as requisições relacionadas ao login e registro de usuários. Ela chama os métodos apropriados na classe LoginModel para executar as operações necessárias.

Alguns dos principais métodos da classe LoginController são:

* login(): Realiza o processo de login do usuário. Ele recebe o email e a senha fornecidos, valida os dados e verifica se o usuário existe no banco de dados. Se o login for bem-sucedido, as permissões do usuário são armazenadas na sessão.
* register(): Realiza o processo de registro de um novo usuário. Ele recebe os dados do formulário, como nome, email, telefone e senha. Antes de registrar o usuário, ele verifica se o email já está em uso. Se a senha atender aos critérios de segurança definidos, o usuário é registrado no banco de dados.
* checkLogin(): Verifica se o usuário está logado. Ele é chamado em determinadas rotas para garantir que o usuário esteja autenticado antes de acessar determinadas páginas ou executar determinadas ações.

A classe LoginModel é responsável por lidar com as operações de banco de dados relacionadas ao login e registro de usuários. Ela utiliza a classe Connection para estabelecer a conexão com o banco de dados.

Alguns dos principais métodos da classe LoginModel são:

* register(): Registra um novo usuário no banco de dados. Ele verifica se o email fornecido já está em uso e, em seguida, insere os dados do usuário na tabela correspondente. Também é realizada uma transação para garantir a consistência dos dados.
* login(): Realiza a autenticação do usuário. Ele verifica se o usuário existe com o email fornecido e, em seguida, verifica se a senha fornecida está correta. Se o login for bem-sucedido, as permissões do usuário são retornadas.
* checkExistingEmail(): Verifica se um email já está em uso por outro usuário. Ele realiza uma consulta no banco de dados para verificar se há algum registro com o email fornecido.

Essa API permite que os usuários realizem o login, registrem-se como novos usuários e verifiquem se estão autenticados. Ela garante a segurança das senhas armazenando-as de forma criptografada no banco de dados.

## ESTRUTURA DO BANCO DE DADOS E RELACIONAMENTOS

Estrutura do Banco de Dados: Descrição das tabelas, relacionamentos e campos do banco de dados utilizado pela aplicação.

### Banco de Dados: Task Manager Versão do Servidor: 8.0.30

### Tabelas:

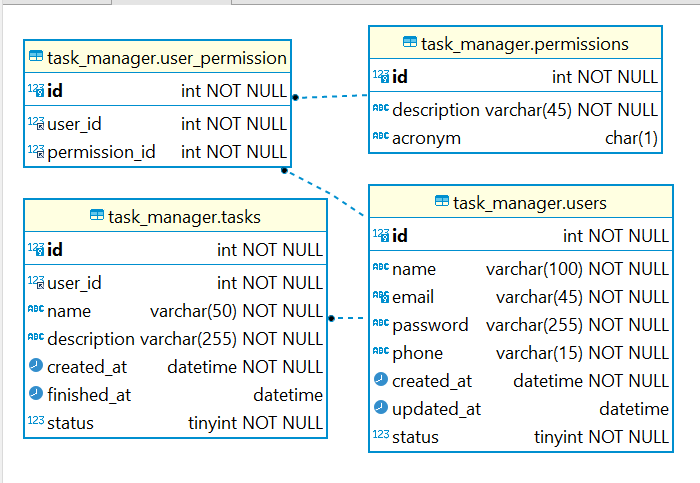
1. Tabela permissions:
   * Descrição: Armazena informações sobre as permissões disponíveis.
   * Colunas:
     + id (int): Identificador único da permissão.
     + description (varchar): Descrição da permissão.
     + acronym (char): Sigla da permissão.
2. Tabela tasks:
   * Descrição: Armazena informações sobre as tarefas.
   * Colunas:
     + id (int): Identificador único da tarefa.
     + user\_id (int): ID do usuário responsável pela tarefa.
     + name (varchar): Nome da tarefa.
     + description (varchar): Descrição da tarefa.
     + created\_at (datetime): Data e hora de criação da tarefa.
     + finished\_at (datetime): Data e hora de conclusão da tarefa (pode ser nulo).
     + status (tinyint): Status da tarefa.
3. Tabela user\_permission:
   * Descrição: Mapeamento entre usuários e permissões.
   * Colunas:
     + id (int): Identificador único da relação usuário-permissão.
     + user\_id (int): ID do usuário.
     + permission\_id (int): ID da permissão.
4. Tabela users:
   * Descrição: Armazena informações sobre os usuários.
   * Colunas:
     + id (int): Identificador único do usuário.
     + name (varchar): Nome do usuário.
     + email (varchar): E-mail do usuário.
     + password (varchar): Senha do usuário.
     + phone (varchar): Número de telefone do usuário.
     + created\_at (datetime): Data e hora de criação do usuário.
     + updated\_at (datetime): Data e hora da última atualização do usuário.
     + status (tinyint): Status do usuário.

### Relacionamentos:

* A tabela tasks possui uma chave estrangeira (user\_id) que faz referência à tabela users (coluna id).
* A tabela user\_permission possui duas chaves estrangeiras: user\_id (referência à tabela users) e permission\_id (referência à tabela permissions).

Essa estrutura de banco de dados permite armazenar informações sobre as permissões disponíveis, tarefas, usuários e os relacionamentos entre eles. É possível associar permissões aos usuários por meio da tabela user\_permission e atribuir tarefas a usuários específicos usando a coluna user\_id na tabela tasks. Isso permite que a aplicação gerencie as permissões dos usuários e acompanhe as tarefas atribuídas a eles.

### Diagrama ER:



## SUPORTE DA APLICAÇÃO:

### Contato: VITOR SAMUEL DO AMARAL PEDROSO

### Número: (35) 9 8416-2931