Faculdade de Design, Tecnologia e Comunicação
Universidade Europeia

Licenciatura em Engenharia Informática | LEIF002D02| 24-25

UC | Projeto de Desenvolvimento Móvel

Docente | Pedro Rosa

Briefing do projeto: Task Master

Autores:

Samuel da Graça Soares Ferreira- 20220755
Edson Sebastião Netchitchi Fernandes Manuel- 20230691
José da Silva Morais - 20230315

Lisboa
20 de Outubro 2024

Introdução

O projeto consiste na implementação de uma aplicação mobile para gerenciamento de tarefas, proporcionando assim aos usuários uma maior organização e um aumento significativo da produtividade dos mesmos.

Nome e Proposta Inicial do Projeto

O Task Master é um sistema de gerenciamento de Tarefas criado exclusivamente para usuários que desejam gerenciar melhor as suas tarefas proporcionando assim um aumento na produtividade. É um aplicativo de gerenciamento de tarefas, que permite a criação e o gerenciamento de tarefas individuais ou em grupo, fornece estatísticas sobre as tarefas executadas e possui um sistema de pontuação e ranking por tarefas concluídas e por tempo de uso da função pomodoro associada a tarefas.

Enquadramento

Problemática

Nos últimos tempos, temos notado que as pessoas têm tido sérios problemas com gerenciamento do seu tempo, o que consequentemente resulta na falta de produtividade e desenvolvimento pessoal e profissional da parte das pessoas. Para uma melhor gestão de tempo e aumento de produtividade é necessário definir o que se pretende fazer, como fazer, e quando fazer. Resumidamente é necessário que se planifiquem as tarefas diárias para um melhor aproveitamento e gerenciamento de tempo.

Objetivos

- **Geral**: implementação e desenvolvimento de uma aplicação de gerenciamento de tarefas.

- **Específicos:**

    1. Ajudar os usuários a gerirem as suas tarefas diárias;
    2. Permitir que os usuários a gerenciem melhor o seu tempo;
    3. Contribuir para que os usuários aumentem o seu nível de produtividade.

Público -alvo

Temos como público-alvo, estudantes que almejam fazer um melhor gerenciamento do seu tempo e das suas tarefas. Além de estudantes, no geral, o nosso público alvo são pessoas que sentem a necessidade de serem mais produtivas, de gerirem melhor o seu tempo e também gerirem melhor as suas tarefas diárias.

Aplicações existentes

Existem 3 aplicações que mais se assemelham aquilo que pretendemos oferecer:

- Tick Tick
- Productive
- Reminders

Guiões

- **Criar uma lista**

  A nossa aplicação dá a possibilidade de criar listas para organização de tarefas.

  Passo:
  1. Abrir o menu lateral;
  2. Clicar no botão de criar listas;
  3. Definir o nome da lista;
  4. Clicar em confirmar.

- **Criar uma tarefa**

  A hierarquia do nosso aplicativo sugere que toda tarefa está vinculada a uma lista, logo, não existem tarefas sem listas.

  Passos:
  1. Clicar no botão de criar tarefa;
  2. Adicionar o nome da tarefa;
  3. Adicionar a data da tarefa (opcional);
  4. Adicionar a descrição da tarefa (opcional);
  5. Adicionar a hora da tarefa (opcional);
  6. Adicionar um tempo para lembrete (opcional);
  7. Clicar em confirmar.

- **Usar o Pomodoro**

  O pomodoro é um cronómetro da aplicação que pode ser vinculado a uma tarefa ou não, fazendo assim com que se criem tarefas cronometradas.

  Passos:
  1. Abrir o menu lateral;
  2. Clicar na opção Pomodoro;
  3. Apertar em iniciar;
  4. Apertar em Break para pausar o cronômetro.

- **Estatísticas**
  A aplicação dá a oportunidade de aceder às estatísticas de uso do usuário. Nas estatísticas constam dados como a quantidade de tarefas realizadas, o tempo de uso do Pomodoro e muito mais.
  Passos:
  1. Abrir o menu lateral;
  2. Clicar em estatísticas.

Casos de uso

Caso de uso 1:

O primeiro caso de uso se resume a criação de uma lista “ Banco de dados”, que será uma lista para guardar tarefas relacionadas a estudar sobre banco de dados, e nesta lista serão adicionadas duas tarefas, “Instalar MySQL” e “Criar o primeiro Banco de dados”, a primeira tarefa (Instalar MySQL), será agendada para o dia “19/11/2024” às “18h30min” e não receberá nenhum lembrete. Já a segunda tarefa (Criar o primeiro Banco de Dados) não receberá nenhuma outra informação a não ser o nome e a descrição “Realizar após a instalação do MySQL”. Para executar este caso de uso são necessários as instruções dos guiões “Criar uma Lista” e “Criar uma Tarefa”.

Caso de uso 2:

O segundo caso de uso está relacionado ao Pomodoro. Supondo que um usuário queira cronometrar e dividir uma sessão de estudo em 3 sessões de 25 min cada, com pausas de 5 min. O usuário pode seguir os passos do guião “Usar o Pomodoro” para assim organizar as suas sessões de estudo.

Enquadramento das unidades curriculares

Neste projeto serão inclusas cinco unidades curriculares, com diferentes funções e aplicações, das

quais podemos destacar:

- Bases de dados: Esta UC destinasse a criação da base de dados que possuirá as informações de todos os usuários, permitindo que eles sejam manipulados conforme a necessidade do funcionamento da app, através da tecnologia SQL.
- Competências Comunicacionais: Permitirá adopção de métodos e técnicas que viabilizarão a melhor apresentação do projeto, tendo em conta as melhores práticas de transmissão de informações.
- Matemática Discreta: Será essencial para fornecer fundamentos teóricos essenciais para a ciência da computação, incluindo conceitos como lógica e álgebra booleana.
- Programação de Dispositivos Móveis: Esta UC é responsável pela a criação de toda a interface da aplicação através do uso da tecnologia Kotlin e Jatpack Compose.
- Programação Orientada a Objetos: Viabilizará a criação dos servidores que permitirão o melhor fluxo das informações, através da utilização de frameworks como Spring Boot.

Plano de Trabalho

Componentes principais

• Frontend (Interface do usuário)

• Backend (Servidor e lógica da app)

• Banco de dados (Armazenamento de tarefas).

Frontend

O frontend da app será construído utilizando o Android Studio, utilizando como linguagem o Kotlin e o framework de auxílio será o JetPack Compose. A interface do usuário permitirá ao mesmo gerenciar as listas de tarefas, tarefas, utilizar o pomodoro e visualizar as estatísticas relacionadas às tarefas e o tempo de uso do modo focus.

Backend

A parte backend, que está relacionada as APIs e ao Servidor, será feito com Java, utilizando o framework Spring Boot. Além das APIs e do servidor, serão também criadas com o Spring Boot as conexões entre o frontend e o servidor, e entre o servidor e o banco de dados, com a implementação de APIs Restful. O backend será o responsável por processar as requisições do frontend e gerenciar as operações com as tarefas. Ele será implementado como uma API simples.

Banco de dados

O banco de dados será implementado com MySQL, usando a linguaguem SQL, e terá a implementação de diversos mecanismos de proteção de dados que virão da parte do Spring Boot. O banco de dados será responsável por armazenar as informações das tarefas e dos usuários. Cada tarefa terá campos como título, descrição, data de vencimento e prioridade.

Requisitos Funcionais e Não Funcionais

Para a execução do projeto os requisitos técnicos estão divididos em dois grupos:

• Requisitos funcionais: Sistema de login, Sistema de cadastro, Gerenciamento de tarefas, Gerenciamento de tarefas em grupo, Pomodoro, Estatísticas das tarefas do utilizador e Lembretes.

• Requisitos não funcionais: Calendário dinâmico, Sistema de ranking por pontuações entre os utilizadores.

Tecnologias a Utilizar

Para o desenvolvimento do projeto serão utilizadas tecnologias como:

• Aplicações: Visual Studio Code, Android Studio, MySQL Workbench, GitHub, Xampp, Excel.

• Websites: Figma, Miro e Monday.

• Linguagens: SQL, Kotlin e Java.

• Frame works: Spring Boot.

Fluxo de dados

Criação de Tarefa:

• O usuário insere os detalhes da tarefa no frontend (título, descrição, data, status).

• O frontend faz uma requisição POST para o backend.

• O backend salva a tarefa no banco de dados.

Listagem de Tarefas:

• O frontend solicita ao backend a lista de tarefas via uma requisição GET.

• O backend recupera as tarefas do banco de dados e retorna para o frontend exibi
las.

Edição de Tarefa:

• O usuário seleciona uma tarefa para edição.

• O frontend envia uma requisição PUT com os novos dados.

• O backend atualiza a tarefa no banco de dados.

Exclusão de Tarefa:

• O usuário seleciona uma tarefa para exclusão.

• O frontend envia uma requisição DELETE.

• O backend remove a tarefa do banco de dados.

Essa arquitetura é simples, mas eficiente para um aplicativo de gerenciamento de tarefas. É escalável, permitindo a adição de funcionalidades como autenticação de usuários ou envio de notificações no futuro. Para este projeto semestral, ela oferece uma base sólida com o foco em CRUD de tarefas, utilizando tecnologias acessíveis e amplamente suportadas.

Grafico de Gantt

O gráfico de Gantt é caracterizado por 6 etapas, dentre elas:

• Planeamento e pesquisa;

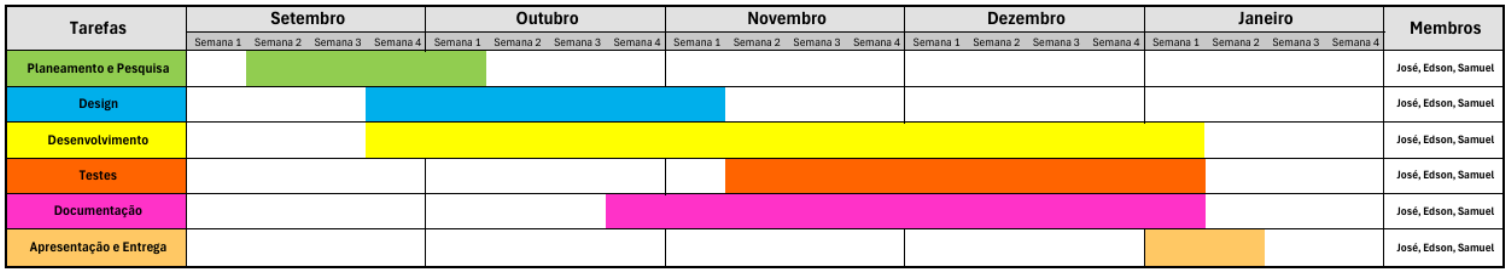
• Design;

• Desenvolvimento;

• Testes;

• Documentação;

• Apresentação e entrega.



Personas

João, o Estudante Universitário Organizado

- **Idade**: 21
- **Profissão**: Estudante de Engenharia de Softwares
- **Objetivos**:
    - Gerenciar projetos acadêmicos e tarefas diárias.
    - Melhorar a produtividade e evitar perder prazos.
    - Organizar suas tarefas por categorias (estudos, lazer, vida pessoal).
- **Comportamento e Frustrações**:
    - Costuma fazer listas em papel ou aplicativos genéricos, mas sente que falta organização.
    - Dificuldade em equilibrar o estudo e momentos de descanso.
    - Precisa de notificações e lembretes para não se esquecer das tarefas.
- **Motivações para usar a app**:
    - A busca por uma ferramenta intuitiva, com lembretes automáticos e integração com o calendário.
    - Quer poder acessar sua agenda em qualquer lugar.

Mariana, a Gestora Atarefada

- **Idade**: 35
- **Profissão**: Gerente de Marketing
- **Objetivos**:
    - Organizar suas responsabilidades de trabalho e vida pessoal em um único lugar.
    - Delegar tarefas para sua equipe de forma eficaz.
    - Visualizar prazos e acompanhar o progresso de projetos.
- **Comportamento e Frustrações**:
    - Utiliza várias ferramentas separadas, o que gera confusão e perda de tempo.
    - Precisa priorizar tarefas em meio a uma rotina agitada e cheia de reuniões.
    - Sente que nem todos os apps oferecem funções de colaboração adequadas.
- **Motivações para usar a app** :
    - Busca por um app que permita criar subtarefas, compartilhar projetos com colegas e definir prioridades claras.
    - Valoriza um layout limpo e profissional.

- **Laura, a freelancer focada**
- **Idade**: 26
- **Profissão**: Designer Gráfica Freelancer
- **Objetivos**:
    - Aumentar o foco e evitar distrações enquanto trabalha em projetos criativos.
    - Gerenciar melhor o tempo para equilibrar trabalho e pausas saudáveis.
    - Cumprir prazos sem se sentir sobrecarregada.
- **Comportamento e frustrações**:
    - Tende a procrastinar e perder foco devido a notificações e redes sociais.
    - Trabalha melhor com blocos de tempo curtos e objetivos claros.
    - Já usou técnicas de pomodoro antes, mas não encontrou uma ferramenta que combine essa funcionalidade com a organização de tarefas.
- **Motivações para usar a app**:
    - Busca uma ferramenta que integre o método pomodoro com a gestão de tarefas, para planejar e executar atividades no mesmo lugar.
    - Deseja notificações suaves para início e fim de ciclos de trabalho e pausas.
    - Gosta de acompanhar relatórios de produtividade para avaliar seu progresso.

Documentação da API

/task-master Explore

Task Master API 1.0 OAS 3.0

/task-master

API para um gestor de tarefas

Contact Edson Manuel

Servers

http://localhost:8080 - Generated server url

Lembretes Tudo relacionado a Lembretes de tarefas

POST / Criar um novo lembrete

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema

```
{
  "id": 0,
  "message": "string",
  "reminderDate": "2024-11-24",
  "reminderTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  }
}
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema


```
{
  "id": 0,
  "message": "string",
  "reminderDate": "2024-11-24",
  "reminderTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  }
}
```


GET /{id}

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

Code	Description	Links


```
{
  "id": 0,
  "message": "string",
  "reminderDate": "2024-11-24",
  "reminderTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  }
}
```


GET /lists

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls Accept header.

Example Value Schema


```
[
  {
    "id": 0,
    "message": "string",
    "reminderDate": "2024-11-24",
    "reminderTime": {
      "hour": 0,
      "minute": 0,
      "second": 0,
      "nano": 0
    }
  }
]
```


GET /delete/{id}

Parameters Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*


```
{
  "id": 0,
  "message": "string",
  "reminderDate": "2024-11-24",
  "reminderTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  }
}
```


Usuarios Tudo relacionado a usuarios

POST `/api/v1/userClients` Criar um novo usuario

Parameters Try it out

No parameters

Request body required application/json

Example Value Schema

```
{
  "username": "string",
  "password": "string",
  "email": "string"
}
```


Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*


```
{
  "id": 0,
  "username": "string",
  "role": "string"
}
```


POST `/api/v1/userClients/updatepassword/{userid}/{newpassword}`

Parameters Try it out

Name	Description
userid * required integer($int64) *(path)*	userid
newpassword * required string *(path)*	newpassword

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "username": "string",
  "role": "string"
}
```

POST `/api/v1/userClients/updatename/{userid}/{newname}`

Parameters Try it out

Name	Description
userid * required `integer($int64)` *(path)*	userid
newname * required `string` *(path)*	newname

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
```

Code	Description	Links

```
  "username": "string",
  "role": "string"
}
```

GET /api/v1/userClients/{id}

Parameters Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "username": "string",
  "role": "string"
}
```

GET /api/v1/userClients/userlist

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*

```
[
  {
    "id": 0,
    "username": "string",
    "role": "string"
  }
]
```

DELETE /api/v1/userClients/delete/{id}

Parameters

Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "username": "string",
```

Code	Description	Links

```
  "role": "string"
}
```

Task List Tudo relacionado a listas de tarefas

POST `/api/v1/tasklists/updatename/{tasklistId}/{newname}`

Parameters Try it out

Name	Description
tasklistId * required `integer($int64)` *(path)*	tasklistId
newname * required `string` *(path)*	newname

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "name": "string",
  "userId": 0
}
```

POST `/api/v1/tasklists/create/{userid}` Criar uma nova lista de tarefas para um usuario

Parameters

Try it out

Name	Description
userid * required `integer($int64)` *(path)*	userid

Request body required application/json

Example Value Schema

```
{
  "name": "string"
}
```

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "name": "string",
  "userId": 0
}
```

GET /api/v1/tasklists/{id}

Parameters Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema { "id": 0, "name": "string", "userId": 0 }	*No links*

GET /api/v1/tasklists/lists

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header.	*No links*

Code	Description	Links

Example Value Schema


```
[
  {
    "id": 0,
    "name": "string",
    "userId": 0
  }
]
```


GET `/api/v1/tasklists/gettasks/{listId}`

Parameters Try it out

Name	Description
listId * required `integer($int64)` *(path)*	listId

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. **Example Value** Schema	*No links*


```
[
  {
    "id": 0,
    "name": "string",
    "priority": "string",
    "dueDate": "2024-11-24",
    "dueTime": {
      "hour": 0,
      "minute": 0,
      "second": 0,
      "nano": 0
    },
    "status": "string",
    "listId": 0
```

Code	Description	Links

```
  }
]
```

DELETE `/api/v1/tasklists/delete/{listId}`

Parameters Try it out

Name	Description
listId * required `integer($int64)` *(path)*	listId

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "name": "string",
  "userId": 0
}
```

Tarefas Tudo relacionado a tarefas

POST `/api/v1/tasks/{listId}` Criar uma nova tarefa em uma lista

Parameters Try it out

Name	Description
listId * required `integer($int64)` *(path)*	listId

Request body required — application/json

Example Value Schema


```
{
  "listId": 0,
  "name": "string",
  "priority": "string",
  "dueDate": "2024-11-24",
  "dueTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  }
}
```


Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema


```
{
  "id": 0,
  "name": "string",
  "priority": "string",
  "dueDate": "2024-11-24",
  "dueTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  },
  "status": "string",
  "listId": 0
}
```


POST `/api/v1/tasks/updatename/{taskId}/{newName}`

Parameters Try it out

Name	Description
taskId * required `integer($int64)` *(path)*	taskId
newName * required `string` *(path)*	newName

Responses

Code	Description	Links
200	OK	*No links*

Media type

Code	Description	Links

/

Controls Accept header.

Example Value Schema


```
{
  "id": 0,
  "name": "string",
  "priority": "string",
  "dueDate": "2024-11-24",
  "dueTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  },
  "status": "string",
  "listId": 0
}
```


GET `/api/v1/tasks/{taskId}`

Parameters Try it out

Name	Description
taskId * required `integer($int64)` *(path)*	taskId

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*


```
{
  "id": 0,
  "name": "string",
  "priority": "string",
```

Code	Description	Links


```
  "dueDate": "2024-11-24",
  "dueTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  },
  "status": "string",
  "listId": 0
}
```


GET /api/v1/tasks/lists

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema


```
[
  {
    "id": 0,
    "name": "string",
    "priority": "string",
    "dueDate": "2024-11-24",
    "dueTime": {
      "hour": 0,
      "minute": 0,
      "second": 0,
      "nano": 0
    },
    "status": "string",
    "listId": 0
  }
]
```


DELETE /api/v1/tasks/delete/{taskId}

Parameters Try it out

Name	Description
taskId * required integer($int64) *(path)*	taskId

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*

```
{
  "id": 0,
  "name": "string",
  "priority": "string",
  "dueDate": "2024-11-24",
  "dueTime": {
    "hour": 0,
    "minute": 0,
    "second": 0,
    "nano": 0
  },
  "status": "string",
  "listId": 0
}
```

Schemas

UserClientCreateDTO

UserClientResponseDTO

LocalTime

TaskCreateDTO

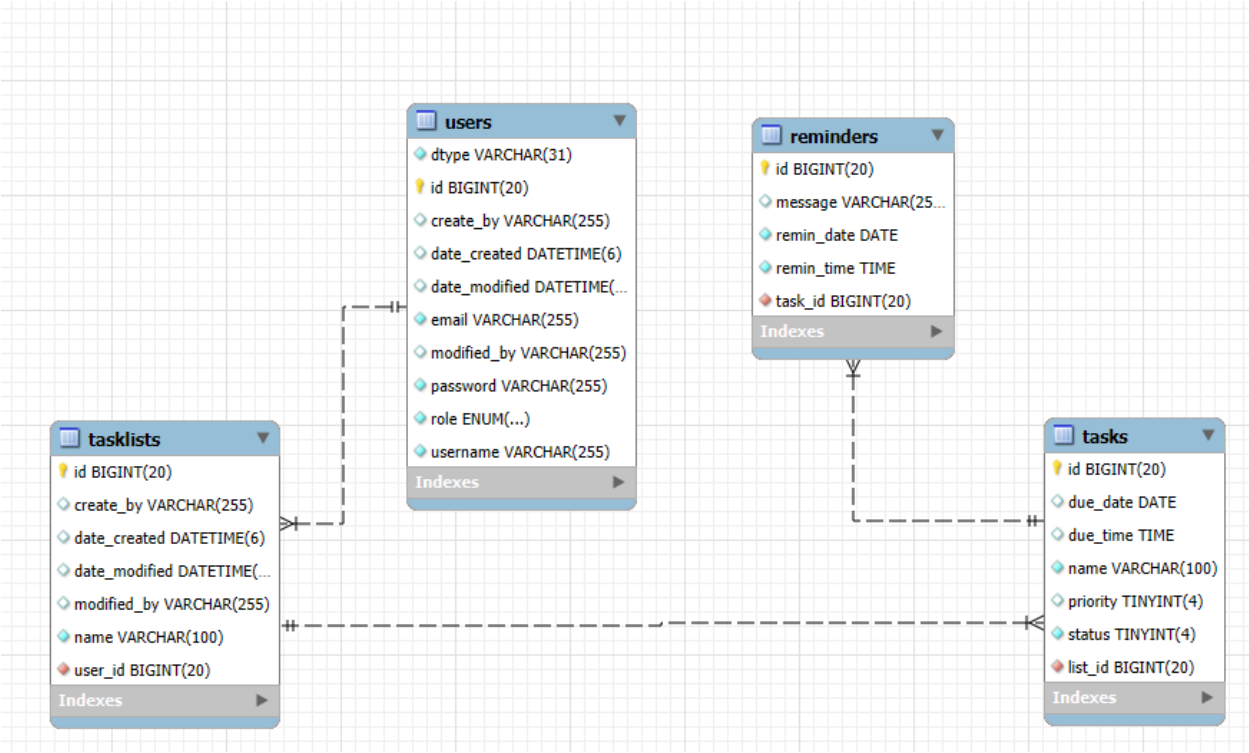
TaskResponseDTO

TaskListResponseDTO

TaskListCreateDTO

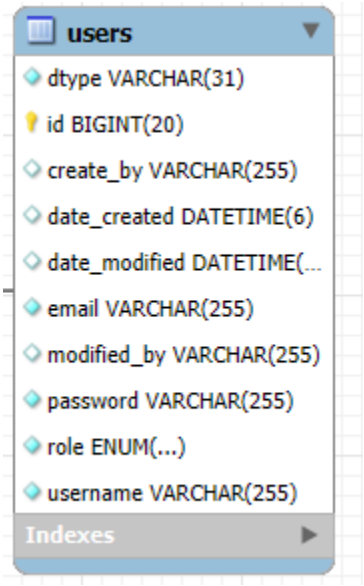
Reminder

Diagrama de classes

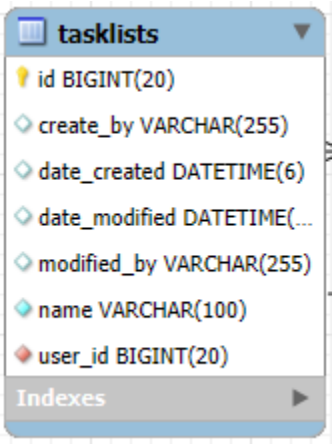


Dicionário de dados

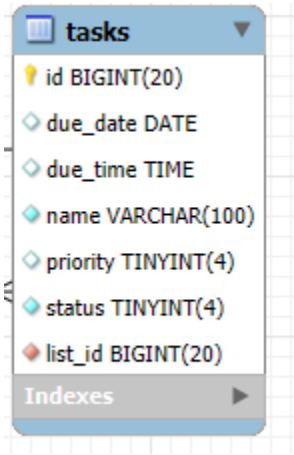
User



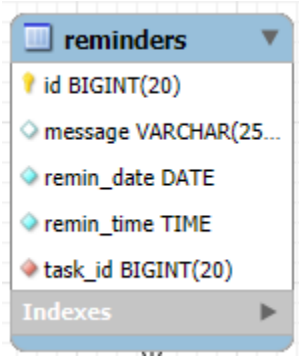
Task List



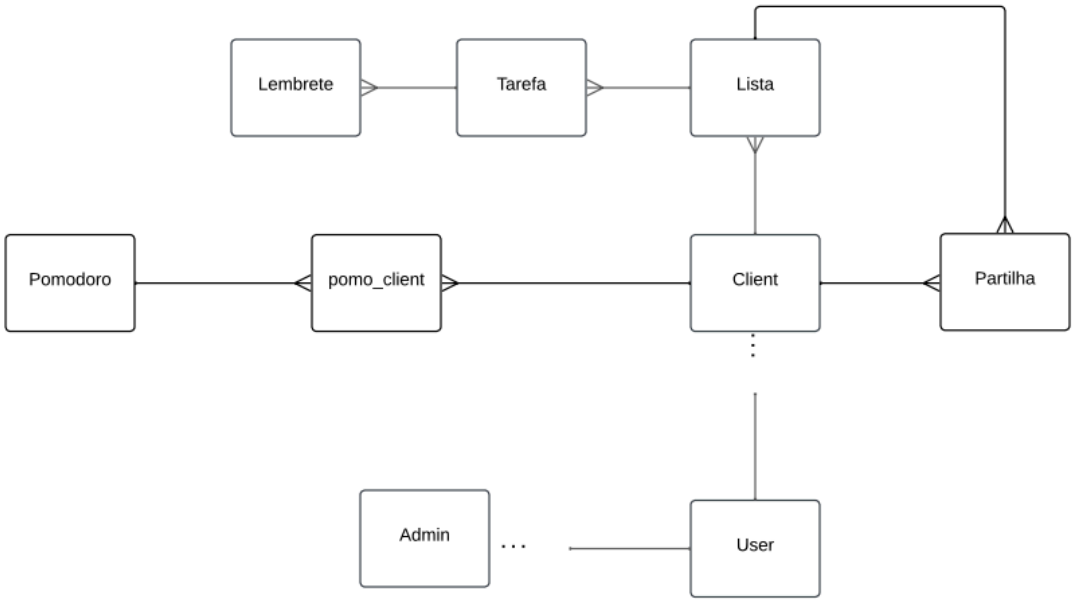
Task



Reminder



MER


Lembrete
Tarefa
Lista
Pomodoro
pomo_client
Client
Partilha
Admin
User

Conclusão

O Task master é um aplicativo de gerenciamento de tarefas, que permite a criação, gerenciamento de tarefas individuais e em grupo, fornece estatísticas sobre as tarefas executadas e possui um sistema de pontuação por tarefa concluída.

Tem como objetivos fornecer uma experiência de usuário simples e direta ao ponto, permitindo que sempre que o usuário entrar na aplicação ele esteja focado nas tarefas que precisa acessar, criar ou editar.

Para isso nos propomos a disponibilizar uma interface simples que centraliza as tarefas, sem muitas informações a serem apresentadas de uma única vez. Almeja também ser capaz de facilitar a organização de tarefas, tendo a possibilidade de agrupar as tarefas em listas, criar tarefas em grupo, definir prioridade e data para a entrega/realização de cada uma das tarefas.

Com o objetivo de incentivar os usuários a serem mais produtivos, criamos um sistema de pontuação para cada tarefa ser realizada, seguindo a estratégia que os aplicativos de exercícios físicos têm utilizado atualmente, como o Fitness da Apple.