Project Manager

Índice

1 Avalidação Técnica

- 1.1 Project Manager
  - 1.1.1 Descrição
  - 1.1.2 Diagrama de Classes

2 Modelos de dados do sistema

- 2.1
- 2.2
- 2.3
  - 2.3.1 Descrição dos atributos
  - 2.3.2
  - 2.3.3
  - 2.3.4
  - 2.3.5
  - 2.3.6
  - 2.3.7
  - 2.3.8
- 2.4
- 2.5
  - 2.5.1
  - 2.5.2
  - 2.5.3
  - 2.5.4
  - 2.5.5
  - 2.5.6

3 Views relacionadas à projetos

- 3.1
  - 3.1.1
  - 3.1.2
  - 3.1.3
  - 3.1.4
  - 3.1.5
- 3.2
  - 3.2.1
  - 3.2.2
  - 3.2.3
- 3.3

- 3.3.1
- 3.3.2
- 3.4
  - 3.4.1
  - 3.4.2
- 3.5
  - 3.5.1
  - 3.5.2
  - 3.5.3
  - 3.5.4
  - 3.5.5

4 Views relacionadas à tasks

- 4.1
  - 4.1.1
  - 4.1.2
  - 4.1.3
  - 4.1.4
  - 4.1.5
  - 4.1.6
- 4.2
  - 4.2.1
  - 4.2.2
  - 4.2.3
- 4.3
  - 4.3.1
  - 4.3.2
- 4.4
  - 4.4.1
  - 4.4.2
- 4.5
  - 4.5.1
- 4.6
  - 4.6.1
  - 4.6.2
  - 4.6.3
  - 4.6.4
  - 4.6.5

5 Views relacionadas à clientes

- 5.1

    - 5.1.1
    - 5.1.2
    - 5.1.3
    - 5.1.4
    - 5.1.5
- 5.2
    - 5.2.1
    - 5.2.2
    - 5.2.3
- 5.3
    - 5.3.1
    - 5.3.2
- 5.4
    - 5.4.1
    - 5.4.2
- 5.5
    - 5.5.1
    - 5.5.2
    - 5.5.3
    - 5.5.4
    - 5.5.5

1 Avalidação Técnica

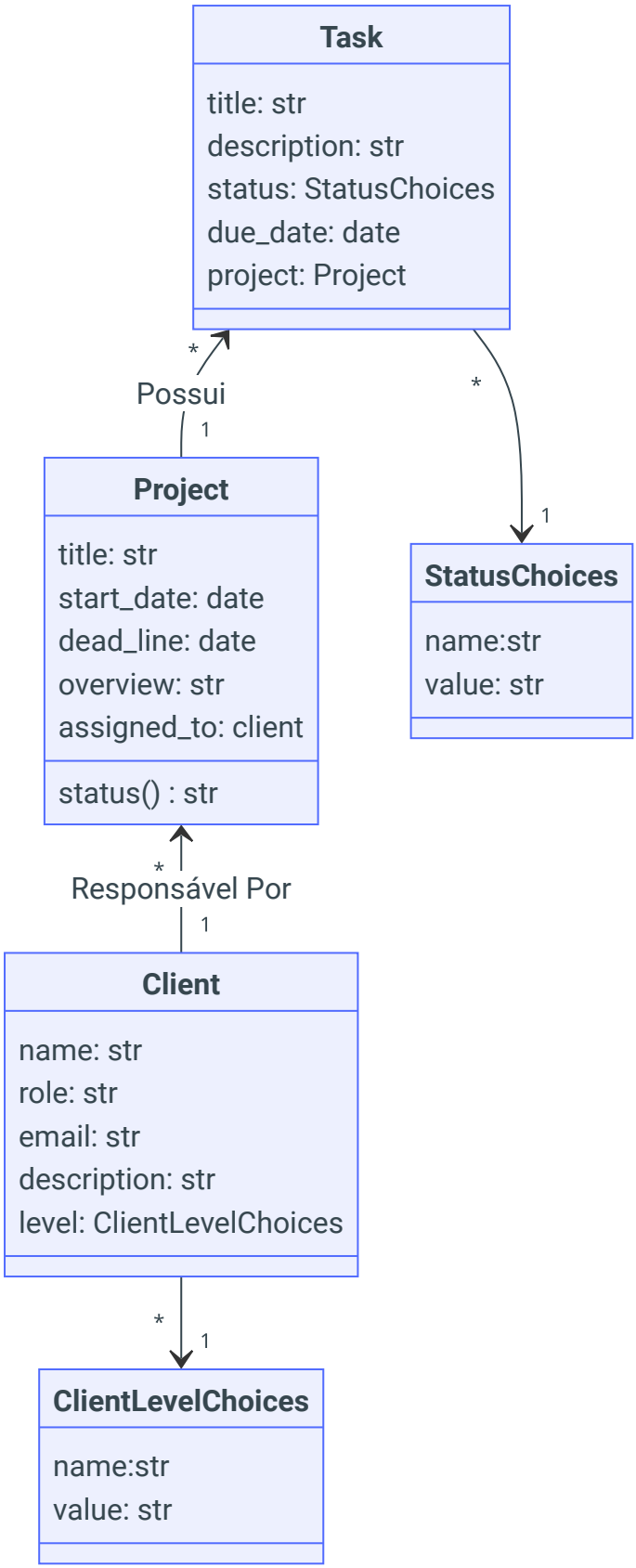
1.1 Project Manager

1.1.1 Descrição

Você foi contratado pela empresa Microsoft para desenvolver a nova plataforma de gestão de projetos e desenvolvimento de software. O sistema legado é desktop e não atende mais às necessidades da empresa, visto que, em épocas de COVID-19 os colaboradores e clientes precisam acessar a plataforma pela internet.

O sistema consiste em gerenciar os projetos da empresa, visando o planejamento e acompanhamento dos times, projetos, clientes, tarefas, entre outros.

1.1.2 Diagrama de Classes


Task
title: str
description: str
status: StatusChoices
due_date: date
project: Project
*
Possui
1
*
1
Project
title: str
start_date: date
dead_line: date
overview: str
assigned_to: client
status() : str
StatusChoices
name:str
value: str
*
Responsável Por
1
Client
name: str
role: str
email: str
description: str
level: ClientLevelChoices
*
1
ClientLevelChoices
name:str
value: str

2 Modelos de dados do sistema

Modelos de dados do app Project.

2.1 Client

Bases: `BaseModel`

Classe mínima para abstração de clientes.

Examples:

Criar uma instância da classe Client

```
>>> from apps.project.models import Client
>>> c:Client = Client.objects.create(
>>> name="Jhon Wayne",
>>> role="Developer",
>>> level=ClientLevelChoices.SENIOR,
>>> email="jhonwaine@acme.com",
>>> description="Melhor dev python")
```

2.2 ClientLevelChoices

Bases: `TextChoices`

Enum para os status do projeto.

2.3 Project

Bases: `BaseModel`

Classe mínima para abstração de projetos

Examples:

Criar uma instância da classe Projet

```
>>> from apps.project.models import Project
>>> import datetime
>>> p = Project.objects.create(
>>> title="Test Project",
>>> project_type=Project.ProjectType.DEVELOPMENT,
>>> start_date=datetime.date(2024, 5, 20),
>>> dead_line=datetime.date(2024, 7, 15),
>>> assigned_to=Client.objects.create(name="Jhon Waine")
>>> overview=overview)
```

2.3.1 Descrição dos atributos

2.3.2 assigned_to `class-attribute` `instance-attribute`

```
assigned_to = ForeignKey(
    Client,
    verbose_name="Responsável do Projeto",
    on_delete=PROTECT,
    null=False,
    blank=False,
    related_name="projects",
)
```

Responsável pelo projeto

2.3.3 dead_line `class-attribute` `instance-attribute`

```
dead_line = DateField('Dead Line', null=False, blank=False)
```

Data de fim estimado do projeto

2.3.4 overview `class-attribute` `instance-attribute`

```
overview = TextField("Visão Geral", null=False, blank=False)
```

Visão geral do projeto

2.3.5 start_date `class-attribute` `instance-attribute`

```
start_date = DateField(
    "Data Inicial", null=False, blank=False
)
```

Data de inicio do projeto

2.3.6 status `property`

```
status: StatusChoices
```

Determina o status do projeto baseado no status das tarefas

2.3.7 title `class-attribute` `instance-attribute`

```
title = CharField(
    "Título do Projeto",
    null=False,
    blank=False,
    max_length=255,
    unique=True,
)
```

Título do projeto

2.3.8 __str__

```
__str__() -> str
```

Representação string do projeto

2.4 StatusChoices

Bases: `TextChoices`

Enum para os status do projeto.

2.5 Task

Bases: `BaseModel`

Classe mínima para abstração das tasks.

Examples:

Criar uma instância da classe Task

```
>>> from apps.project.models import Client
>>> import datetime
>>> t:Task = Task.objects.create(
>>> title="Tarefa 1",
>>> description="Implementação do scrap da página do Yahoo",
>>> status=StatusChoices.TODO,
>>> due_date=datetime.date(2024, 5, 20),
>>> project=Project.objects.first()
>>> )
```

2.5.1 description `class-attribute` `instance-attribute`

```
description = TextField(
    "Descrição", null=False, blank=False
)
```

Descrição da tarefa

2.5.2 due_date `class-attribute` `instance-attribute`

```
due_date = DateField(
    "Data Estimada do Fim", null=False, blank=False
)
```

Data prevista para término da tarefa

2.5.3 project `class-attribute` `instance-attribute`

```
project = ForeignKey(
    Project,
    verbose_name="Projeto",
    on_delete=CASCADE,
    null=False,
    blank=False,
    related_name="tasks",
)
```

Projeto no qual esta tarefa está atrelada

2.5.4 status `class-attribute` `instance-attribute`

```
status = CharField(
    "Status",
    max_length=1,
    null=False,
    blank=False,
    choices=StatusChoices,
    default=TODO,
)
```

Estatus da tarefa

2.5.5 title `class-attribute` `instance-attribute`

```
title = CharField(
    "Título da Tarefa",
    null=False,
    blank=False,
    max_length=255,
    unique=True,
)
```

Título da tarefa

2.5.6 __str__

```
__str__()
```

Representação string da tarefa

3 Views relacionadas à projetos

Views do app Project.

3.1 ProjectCreateView

Bases: `DefaultContextMixin`, `CreateView`

Django Class Based View para criação de um projeto

3.1.1 fields `class-attribute` `instance-attribute`

```
fields = (
    "title",
    "overview",
    "assigned_to",
    "start_date",
    "dead_line",
)
```

Campos a serem manipulados

3.1.2 model `class-attribute` `instance-attribute`

```
model = Project
```

Modelo que esta view manipula

3.1.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Projeto Criado com Sucesso!'
```

Mensagem de Sucesso

3.1.4 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:project.list')
```

URL de Sucesso

3.1.5 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML

3.2 ProjectDeleteView

Bases: `DefaultContextMixin`, `DeleteView`

Django Class Based View para remoção de dados dos Projetos

3.2.1 model `class-attribute` `instance-attribute`

```
model = Project
```

Modelo que esta view manipula

3.2.2 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:project.list')
```

URL de Sucesso

3.2.3 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/delete.html'
```

Nome do template utilizado para renderização do HTML

3.3 ProjectDetailView

Bases: `DefaultContextMixin`, `DetailView`

Django Class Based View para visualização detalhada dos dados de um projeto

3.3.1 model `class-attribute` `instance-attribute`

```
model = Project
```

Modelo que esta view manipula

3.3.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/detail.html'
```

Nome do template utilizado para renderização do HTML

3.4 ProjectListView

Bases: `DefaultContextMixin`, `ListView`

Django Class Based View para listagem de projetos

3.4.1 model `class-attribute` `instance-attribute`

```
model = Project
```

Modelo que esta view manipula

3.4.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/list.html'
```

Nome do template utilizado para renderização do HTML

3.5 ProjectUpdateView

Bases: `DefaultContextMixin`, `UpdateView`

Django Class Based View para atualização dos dados de um projeto

3.5.1 fields `class-attribute` `instance-attribute`

```
fields = (
    "title",
    "overview",
    "assigned_to",
    "start_date",
    "dead_line",
)
```

Campos a serem manipulados

3.5.2 model `class-attribute` `instance-attribute`

```
model = Project
```

Modelo que esta view manipula

3.5.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Projeto Criado com Sucesso!'
```

Mensagem de Sucesso

3.5.4 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:project.list')
```

URL de Sucesso

3.5.5 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML

4 Views relacionadas à tasks

Views do app Task.

4.1 TaskCreateView

Bases: `DefaultContextMixin`, `CreateView`

Django Class Based View para criação de uma tarefa

4.1.1 fields `class-attribute` `instance-attribute`

```
fields = (
    "title",
    "status",
    "due_date",
    "project",
    "description",
)
```

Campos a serem manipulados

4.1.2 model `class-attribute` `instance-attribute`

```
model = Task
```

Modelo que esta view manipula

4.1.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Tarefa Cadastrado Com Sucesso!'
```

Mensagem de Sucesso

4.1.4 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML

4.1.5 get_initial

```
get_initial()
```

Obtém dados para inicialização do formulário de cadastr de uma task

4.1.6 get_success_url

```
get_success_url()
```

Definie a URL de Sucesso de acordo com regras do negócio

4.2 TaskDeleteView

Bases: `DefaultContextMixin`, `DeleteView`

Django Class Based View para remoção de uma tarefa

4.2.1 model `class-attribute` `instance-attribute`

```
model = Task
```

Modelo que esta view manipula

4.2.2 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:task.list')
```

URL de Sucesso

4.2.3 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/delete.html'
```

Nome do template utilizado para renderização do HTML

4.3 TaskDetailView

Bases: `DefaultContextMixin`, `DetailView`

Django Class Based View para visualização de detalhes de uma tarefa

4.3.1 model `class-attribute` `instance-attribute`

```
model = Task
```

Modelo que esta view manipula

4.3.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/detail.html'
```

Nome do template utilizado para renderização do HTML

4.4 TaskListView

Bases: `DefaultContextMixin`, `ListView`

Django Class Based View para listagem de tarefas

4.4.1 model `class-attribute` `instance-attribute`

```
model = Task
```

Modelo que esta view manipula

4.4.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/list.html'
```

Nome do template utilizado para renderização do HTML

4.5 TaskStatusUpdateView

Bases: `DefaultContextMixin`, `View`

Altera o status da tarefa

4.5.1 get

```
get(*args, **kwargs)
```

Recebe o novo status da tarefa via GET e atualiza o statua da intância da tarefa

4.6 TaskUpdateView

Bases: `DefaultContextMixin`, `UpdateView`

Django Class Based View para atualização de uma tarefa

4.6.1 fields `class-attribute` `instance-attribute`

```
fields = (
    "title",
    "status",
    "due_date",
    "project",
    "description",
)
```

Campos a serem manipulados

4.6.2 model `class-attribute` `instance-attribute`

```
model = Task
```

Modelo que esta view manipula

4.6.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Tarefa Atualizado Com Sucesso!'
```

Mensagem de Sucesso

4.6.4 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:task.list')
```

URL de Sucesso

4.6.5 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML

5 Views relacionadas à clientes

Views do app Client.

5.1 ClientCreateView

Bases: `DefaultContextMixin`, `CreateView`

Django Class Based View para criação de Clientes

5.1.1 fields `class-attribute` `instance-attribute`

```
fields = ('name', 'role', 'level', 'email', 'description')
```

Campos a serem manipulados

5.1.2 model `class-attribute` `instance-attribute`

```
model = Client
```

Modelo que esta view manipula

5.1.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Cliente Cadastrado Com Sucesso!'
```

Mensagem de Sucesso

5.1.4 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:client.list')
```

URL de Sucesso

5.1.5 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML

5.2 ClientDeleteView

Bases: `DefaultContextMixin`, `DeleteView`

Django Class Based View para remoção de um cliente

5.2.1 model `class-attribute` `instance-attribute`

```
model = Client
```

Modelo que esta view manipula

5.2.2 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:client.list')
```

Mensagem de Sucesso

5.2.3 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/delete.html'
```

Nome do template utilizado para renderização do HTML

5.3 ClientDetailView

Bases: `DefaultContextMixin`, `DetailView`

Django Class Based View para visualização de detalhes de um cliente

5.3.1 model `class-attribute` `instance-attribute`

```
model = Client
```

Modelo que esta view manipula

5.3.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/detail.html'
```

Nome do template utilizado para renderização do HTML

5.4 ClientListView

Bases: `DefaultContextMixin`, `ListView`

Django Class Based View para listagem de clientes

5.4.1 model `class-attribute` `instance-attribute`

```
model = Client
```

Modelo que esta view manipula

5.4.2 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/list.html'
```

Nome do template utilizado para renderização do HTML

5.5 ClientUpdateView

Bases: `DefaultContextMixin`, `UpdateView`

Django Class Based View para atualização de dados dos clientes

5.5.1 fields `class-attribute` `instance-attribute`

```
fields = ('name', 'role', 'level', 'email', 'description')
```

Campos a serem manipulados

5.5.2 model `class-attribute` `instance-attribute`

```
model = Client
```

Modelo que esta view manipula

5.5.3 success_message `class-attribute` `instance-attribute`

```
success_message = 'Cliente Atualizado Com Sucesso!'
```

Mensagem de Sucesso

5.5.4 success_url `class-attribute` `instance-attribute`

```
success_url = reverse_lazy('project:client.list')
```

URL de Sucesso

5.5.5 template_name `class-attribute` `instance-attribute`

```
template_name = 'core/crud/form.html'
```

Nome do template utilizado para renderização do HTML