Curso Desenvolvedor Backend – Futuro Digital

Professor: Ricardo Luis dos Santos

Aluno: _____

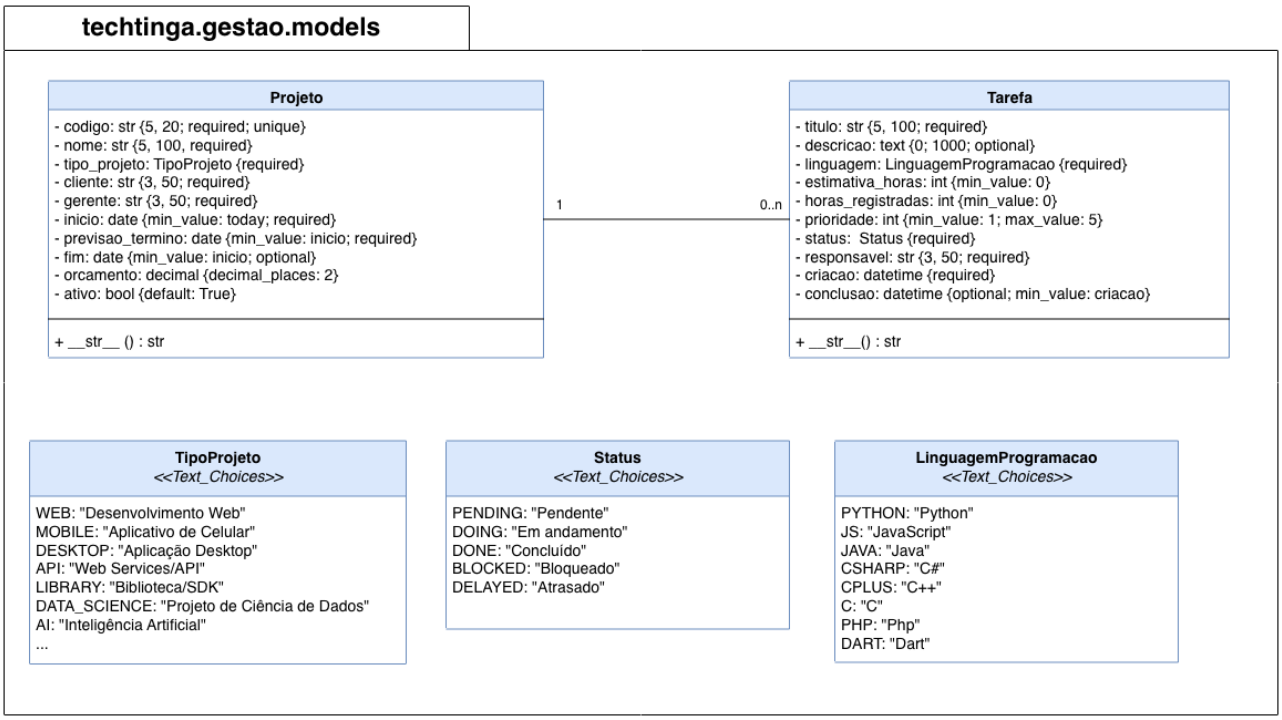
Data: 19.01.2026

DESAFIO FINAL

A empresa de tecnologia chamada **Techtinga** nasceu como uma s*oftware house* enxuta no bairro Restinga, em Porto Alegre, e rapidamente se organizou para gerenciar de múltiplos contratos de desenvolvimento de *software* simultaneamente. Para isso, toda demanda vira um **Projeto** bem definido: cada projeto recebe um **código** único (por exemplo, PRJ-2026-001), um **nome** claro (como "Sistema Web para Gerenciamento Financeiro"), um **tipo_projeto** padronizado via enum (**TipoProjeto** como WEB, MOBILE, API, DATA_SCIENCE ou AI), e já sai "fechado" com **cliente** e **gerente** responsáveis (ex.: cliente "Prefeitura de Porto Alegre" e gerente "Mariana Souza"). A operação também depende de prazos e governança: o projeto registra **inicio** (ex.: 2026-02-01), **previsao_termino** obrigatória e posterior ao início (ex.: 2026-05-30), **fim**, ou seja, a data real de conclusão do projeto (ex.: 2026-05-28) e um **orcamento** com duas casas decimais (ex.: 185000.00). Para manter o portfólio saudável, a Techtinga marca se o projeto está **ativo** (por padrão True) e usa o método __str__ para exibir o projeto de forma legível no portal administrativo, relatórios e APIs (por exemplo, "PRJ-2026-001 — Portal de Serviços do Cidadão").

No dia a dia, cada **Projeto** desdobra em várias **Tarefas** (relação 1 → 0..n), que são a unidade de execução do time. Uma tarefa tem **titulo** (ex.: "Implementar autenticação com token"), **descricao** opcional (ex.: "Bearer token no DRF, expiração e renovação"), e é classificada por **linguagem** via enum (**LinguagemProgramacao** como PYTHON, JS, JAVA, CSHARP, CPLUSPLUS, C, PHP, DART). O planejamento acontece com **estimativa_horas** (ex.: 24) e o acompanhamento com **horas_registradas** (ex.: 18), sempre evitando valores negativos; a criticidade entra em **prioridade** (1 a 5, ex.: 5 para algo bloqueante) e o fluxo é governado por **status** (**Status** como PENDING, DOING, DONE, BLOCKED, DELAYED). A gestão também cobra rastreabilidade: toda tarefa tem **responsavel** (ex.: "João Lima"), **criacao** obrigatória (ex.: 2026-02-03 09:12) e **conclusao** opcional, mas nunca anterior à criação (ex.: 2026-02-06 17:40 quando finalizada). Assim, num mesmo projeto, é comum ver tarefas como "Criar endpoint /projetos" (PYTHON, prioridade 4, DOING) convivendo com "Tela de dashboard do gerente" (JS, prioridade 3, PENDING) e "Pipeline de dados para relatório mensal" (PYTHON, tipo de projeto DATA_SCIENCE, DONE), tudo padronizado para virar API no Django REST Framework e, quando necessário, ser carregado em lote a partir de um CSV fornecido pelo cliente ou pela própria equipe. Note que o arquivo CSV é fornecido e está disponível na tarefa.

Considerando essas informações e o diagrama de classes abaixo. Crie as implementações para modelos considerando o Python 3, o Django 5, Django Rest Framework, a arquitetura MVT, bem como todas as boas práticas e conceitos vistos durante o componente curricular. Lembre-se que todas as implementações devem ser funcionais e inclua **o app no seu projeto**, bem como os modelos no **app django-admin**. Ademais, considere as seguintes especificações:



a. Configuração básica a ser criada:

i. Nome do projeto: **techtinga**

ii. Nome do app: **gestao**

iii. Drive do banco de dados: **sqlite3**

iv. Nome do banco de dados: **banco_techtinga.sqlite3**

b. **(1 ponto)** Implementar o modelo para a classe **Projeto**.

i. Implementar os atributos com as devidas restrições detalhadas no diagrama

ii. Implementar as relações entre os modelos e a cardinalidade de forma correta, quando necessário

iii. Permitir no retorno do método **__str__** a apresentação dos atributos **cliente, nome, tipo_projeto, orcamento, inicio (DD/MM/YYYY) e previsao_termino (DD/MM/YYYY)**.

iv. Disponibilizar o modelo para acesso no **django-admin**

c. **(1 ponto)** Implementar o modelo para a classe **Tarefa**.

i. Implementar os atributos com as devidas restrições detalhadas no diagrama

ii. Implementar as relações entre os modelos e a cardinalidade de forma correta, quando necessário

iii. Permitir no retorno do método **__str__** a apresentação dos atributos **responsavel, linguagem, titulo, criacao (DD/MM/YYYY), conclusao (DD/MM/YYYY), estimativa_horas e status.**

iv. Disponibilizar o modelo para acesso no **django-admin**

d. **(1 ponto)** No *model* **Tarefa**, criar um *validator* para impedir a criação de uma **Tarefa** com data anterior (considerando o campo **criacao)** em relação ao campo **inicio** do model **Projeto.** Além disso, impedir a criação de novas tarefas caso o projeto esteja fechado (**ativo == False**, ou com **data fim** já informada).

e. **(1 ponto)** Implementar um script para realizar a importação dos dados do arquivo CSV em anexo. Note que linhas que não possuam informações obrigatórias devem ser desconsideradas. Ainda, o arquivo contém diversas tarefas para os projetos disponíveis, no entanto, não estão ordenadas.

f. **(2 pontos)** Implementar os métodos de consulta nos *managers* conforme abaixo. Observe que você deverá implementar a consulta pelo modelo no qual está implementando a consulta (não forçar dependências) e validar os dados dos parâmetros. Ainda, você deverá implementar o *package* **managers** no *app* **gestao**.

i. Implementar o método **listar_projetos_linguagem**, o qual ao receber duas datas informadas e uma linguagem de programação deverá consultar os projetos que possuam tarefas iniciadas durante esse período e que tenham sido realizadas na linguagem de programação informada;

listar_projetos_linguagem (inicio: date, fim: date, linguagem: LinguagemProgramacao) -> QuerySet[Projeto]

ii. Implementar o método **listar_tarefas_projeto**, o qual ao receber um projeto deverá consultar as tarefas associadas ao projeto informado. Note que as tarefas devem estar ordenadas por ondem responsável, status e, por fim, data de criação;

listar_tarefas_projeto (projeto: Projeto) -> QuerySet[Tarefa]

iii. Implementar o método **listar_tarefas_colaborador**, o qual ao receber o nome de um responsável, deverá listar as tarefas atribuídas a esse colaborador, agrupadas por status e ordenadas por data de criação;

listar_tarefas_colaborador (responsavel: string) -> QuerySet[Tarefa]

g. **(1 ponto)** Implementar os *web services* completos para que seja possível utilizar qualquer consulta desenvolvida na **letra e** e retornar os registros de acordo com a consulta. O *endpoint* devem estar configurado de acordo com os web services RESTful e utilizar os métodos do protocolo HTTP, de acordo com o visto em sala de aula (neste caso, apenas o método GET). Ainda, deve ser disponibilizado o *endpoint* no repositório.

h. **(2 pontos)** Implementar os web services completos para que seja possível criar, atualizar, exibir um registro, deletar e listar todos os registros de um modelo específico, os *endpoints* devem estar configurados de acordo com os web services RESTful e utilizar os métodos do protocolo HTTP, de acordo com o visto em sala de aula. Além disso, deve ser utilizado pelo menos um método de autenticação e as permissões padrão do Django (utilizando **token**). Ainda, deve ser disponibilizado os *endpoints* no repositório conforme exemplo em sala.

i. **(2 pontos)** O gerente deseja consultar o relatório das tarefas desenvolvidas durante o ano atual. Para que isso seja possível, você deverá implementar um *web service* para retornar as informações necessárias para o preenchimento de um relatório das tarefas desenvolvidas durante o ano atual. O relatório deve apresentar os projetos desenvolvidos durante o ano e um totalizador das tarefas por **status** para cada projeto que possua alguma atividade desenvolvida durante o ano atual. Por fim, apresente o totalizador das horas estimadas e horas registradas para cada **status**. Seu *web service* precisa apresentar os objetos conforme abaixo, contendo as informações do projeto e para cada projeto, os totalizadores de tarefas e as informações de cada tarefa. Lembre-se de que você precisa informar todas essas informações para uma consulta que apenas informa o ano desejado.

Relatório de Tarefas Desenvolvidas — 2025

Projeto: Desenvolvimento do sistema ads.restinga.ifrs.edu.br

Início: 10/09/2025 Previsão Término: 23/12/2025 Orçamento: R$ 500,00

Tarefas Concluídas: 2 | Horas estimadas: 20 horas | Horas registradas: 28 horas

Ricardo Luis dos Santos | Python | Implementação do cadastro no site | 20/09/2025 | 23/09/2025| 15 horas estimadas | 10 horas registradas

Rafael Esteves | JavaScript | Front-end | 10/11/2025 | 13/11/2025| 5 horas estimadas | 13 horas registradas

Tarefas Atrasadas: 1 | Horas estimadas: 15 horas | Horas registradas: 20 horas

Ricardo Luis dos Santos | Python | Geração de Relatórios | 10/10/2025 | 12/10/2025| 15 horas estimadas | 20 horas registradas

Projeto: Aplicativo Android para jogos educativos

Início: 20/11/2025 Previsão Término: 31/12/2025 Orçamento: R$ 50000,00

Tarefas Concluídas: 1 | Horas estimadas: 11 horas | Horas registradas: 30 horas

Rafael Esteves | Dart | Front-end | 10/11/2025 | 13/11/2025| 11 horas estimadas | 30 horas registradas

Abaixo segue um exemplo de objeto JSON que deve ser gerado para que seja possível gerar o relatório.


```json
{
  "ano": 2025,
  "projetos": [
    {
      "nome": "Desenvolvimento do sistema ads.restinga.ifrs.edu.br",
      "inicio": "2025-09-10",
      "previsao_termino": "2025-12-23",
      "orcamento": "500.00",
      "resumo": {
        "tarefas_concluidas": {
          "quantidade": 2,
          "horas_estimadas": 20,
          "horas_registradas": 28,
          "tarefas": [
            {
              "titulo": "Implementação do cadastro no site",
              "linguagem": "PYTHON",
              "estimativa_horas": 15,
              "horas_registradas": 10,
              "status": "DONE",
              "responsavel": "Ricardo Luis dos Santos",
              "criacao": "2025-09-20T00:00:00",
              "conclusao": "2025-09-23T00:00:00"
            },
            {
              "titulo": "Front-end",
              "linguagem": "JS",
              "estimativa_horas": 5,
              "horas_registradas": 13,
              "status": "DONE",
              "responsavel": "Rafael Esteves",
              "criacao": "2025-11-10T00:00:00",
              "conclusao": "2025-11-13T00:00:00"
            }
          ]
        },
        "tarefas_atrasadas": {
          "quantidade": 1,
          "horas_estimadas": 15,
          "horas_registradas": 20,
          "tarefas": [
            {
              "titulo": "Geração de Relatórios",
              "linguagem": "PYTHON",
              "estimativa_horas": 15,
              "horas_registradas": 20,
              "status": "DELAYED",
              "responsavel": "Ricardo Luis dos Santos",
              "criacao": "2025-10-10T00:00:00",
              "conclusao": "2025-10-12T00:00:00"
            }
          ]
        }
      }
    },
    {
      "nome": "Aplicativo Android para jogos educativos",
      "inicio": "2025-11-20",
      "previsao_termino": "2025-12-31",
      "orcamento": "50000.00",
      "resumo": {
        "tarefas_concluidas": {
          "quantidade": 1,
          "horas_estimadas": 11,
          "horas_registradas": 30,
          "tarefas": [
            {
              "titulo": "Front-end",
              "linguagem": "DART",
              "estimativa_horas": 11,
              "horas_registradas": 30,
              "status": "DONE",
              "responsavel": "Rafael Esteves",
              "criacao": "2025-11-10T00:00:00",
              "conclusao": "2025-11-13T00:00:00"
            }
          ]
        },
        "tarefas_atrasadas": {
          "quantidade": 0,
          "horas_estimadas": 0,
          "horas_registradas": 0,
          "tarefas": []
        }
      }
    }
  ]
}
```