UNIVERSIDADE DA MAIA UMAIA

DomusShelf - Sistema de Gestão de Farmácia Doméstica

Guião para Aprovação

Miguel Ângelo Ascensão Real

Aluno UMAIA nº 48891, Disciplina Arquitetura e Desenho de Software, 14 de Janeiro de 2026

INTRODUÇÃO

Contextualização

A gestão de medicamentos no ambiente doméstico representa um desafio quotidiano para muitas famílias. A acumulação de medicamentos com diferentes datas de validade, a dificuldade em controlar o stock disponível e a falta de visibilidade sobre produtos prestes a expirar conduzem frequentemente ao desperdício de recursos e, em casos mais graves, ao consumo inadvertido de medicamentos fora de prazo.

Segundo dados da Ordem dos Farmacêuticos, estima-se que uma percentagem significativa dos medicamentos adquiridos em Portugal acaba por não ser utilizada dentro do prazo de validade. Este cenário evidencia a necessidade de ferramentas que auxiliem os cidadãos na gestão eficaz da sua farmácia caseira.

O projecto DomuShelf surge como resposta a esta problemática, propondo o desenvolvimento de uma aplicação web que permita ao utilizador registar, monitorizar e gerir o stock de medicamentos que possui em casa, com especial enfoque no controlo de validades através de um sistema de alertas proactivo.

OBJECTIVOS DO TRABALHO

O objectivo geral deste trabalho é conceber, desenhar e implementar um protótipo funcional de um sistema de gestão de farmácia doméstica, aplicando princípios de arquitectura e design de software adequados à complexidade do problema.

Objectivos Gerais:

- Desenvolver uma aplicação web funcional com arquitectura clara e bem documentada;
- Implementar um modelo de dados relacional que suporte as operações de gestão de stock;
- Criar uma interface de utilizador responsiva, optimizada para dispositivos móveis;
- Garantir que a aplicação seja facilmente instalável e executável em qualquer ambiente.

Objectivos Específicos:

- Implementar funcionalidades CRUD (Create, Read, Update, Delete) para gestão de medicamentos;
- Desenvolver sistema de gestão de stock por embalagem/lote com controlo de quantidades;
- Criar mecanismo de registo de consumos que actualize automaticamente o stock;
- Implementar sistema de alertas de validade configurável pelo utilizador;
- Desenvolver dashboard informativo com visão geral do estado da farmácia;
- Documentar a arquitectura e produzir manual de utilizador.

REVISÃO CONCEPTUAL E TECNOLÓGICA

Arquitectura Web e Padrão MVC/MTV

A arquitectura web moderna assenta tipicamente no modelo cliente-servidor, onde um navegador (cliente) comunica com um servidor através do protocolo HTTP, enviando pedidos e recebendo respostas. Este modelo de comunicação *request-response* é síncrono e stateless, o que significa que cada pedido é independente e contém toda a informação necessária para o seu processamento.

O padrão arquitectural MVC (Model-View-Controller) estabelece uma separação clara de responsabilidades: o "Model" gere os dados e a lógica de negócio; a "View" apresenta a informação ao utilizador; e o "Controller" processa os pedidos e coordena a interacção entre Model e View.

A framework Django, proposta neste projecto, adopta uma variante denominada MTV (Model-Template-View), onde o "Template" corresponde à "View" do MVC e a "View" do Django corresponde ao "Controller". Esta nomenclatura, embora diferente, mantém os mesmos princípios de separação de responsabilidades.

Framework Django

Django é uma framework web de alto nível escrita em Python que promove o desenvolvimento rápido e o design limpo e pragmático. Criada em 2005 e mantida pela Django Software Foundation, é uma das frameworks web mais populares e maduras do ecossistema Python.

Justificação da escolha:

- Um All-in-one: Django inclui nativamente ORM, sistema de templates, autenticação, administração, e gestão de formulários;
- ORM robusto: Permite definir modelos de dados em Python e abstrair completamente as queries SQL;
- Django Admin: Interface de administração gerada automaticamente, útil para debug e gestão de dados;
- Simplicidade de deployment: Com SQLite, a aplicação corre com um único comando sem dependências externas;
- Documentação extensa: A documentação oficial é considerada uma das melhores entre frameworks web;

Base de Dados SQLite

SQLite é um motor de base de dados relacional que, ao contrário de sistemas como PostgreSQL ou MySQL, não requer um servidor separado. A base de dados é armazenada num único ficheiro, o que simplifica drasticamente a instalação, configuração e portabilidade da aplicação.

Vantagens para este projecto:

- Zero configuração: Não é necessário instalar nem configurar serviços adicionais;
- Portabilidade: O ficheiro .sqlite3 pode ser copiado para qualquer máquina;
- Adequação ao cenário: Para uma aplicação de utilizador único com volume de dados reduzido, SQLite oferece desempenho mais que suficiente;
- Integração nativa: Django suporta SQLite nativamente sem bibliotecas adicionais.

Interface Responsiva com Bootstrap

Bootstrap é uma framework CSS que fornece componentes pré-estilizados e um sistema de grelha responsivo. A abordagem *mobile-first* do Bootstrap garante que a interface seja optimizada primariamente para dispositivos móveis, adaptando-se progressivamente a ecrãs maiores.

A utilização de Bootstrap via CDN (Content Delivery Network) elimina a necessidade de compilar CSS localmente, simplificando o processo de desenvolvimento e garantindo que a aplicação utilize sempre a versão mais recente da biblioteca

ARQUITECTURA DO SISTEMA

Visão Geral

O sistema DomusShelf implementa uma arquitectura web monolítica baseada no padrão MTV (Model-Template-View) do Django. Esta arquitectura foi escolhida pela sua simplicidade, adequação ao âmbito do projecto (aplicação de utilizador único), e facilidade de deployment.

A aplicação é composta por uma única instância Django que serve simultaneamente como servidor web e processador de lógica de negócio. O utilizador interage com a aplicação através de um navegador web, que comunica com o servidor via protocolo HTTP.

Diagrama de Arquitectura:

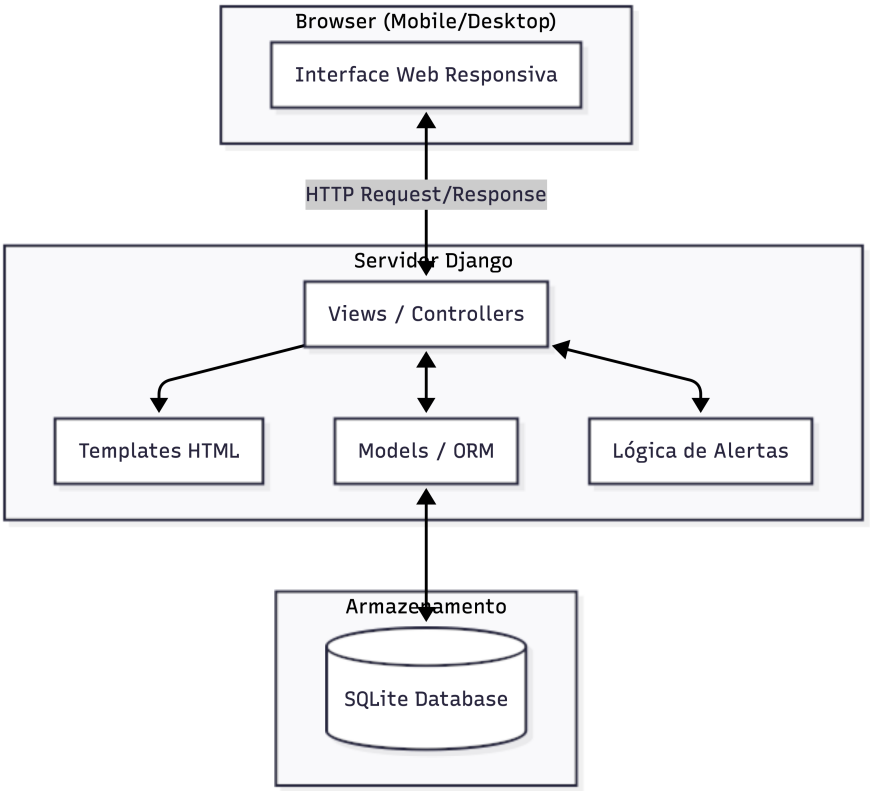

Figura 1 – Arquitectura da Solução

CAMADAS DA APLICAÇÃO

Camada de Apresentação (Templates):

Responsável pela renderização da interface de utilizador. Utiliza o sistema de templates do Django com herança de templates (template inheritance) para manter consistência visual. O template base define a estrutura comum (navbar, footer, estilos) e os templates específicos herdam e estendem esta estrutura.

Camada de Controlo (Views):

Processa os pedidos HTTP recebidos, executa a lógica de negócio necessária, interage com os Models para obter ou modificar dados, e devolve a resposta apropriada (tipicamente um template renderizado). As Views implementam as operações CRUD e a lógica de cálculo de alertas.

Camada de Dados (Models):

Define a estrutura dos dados e encapsula o acesso à base de dados através do ORM do Django. Os Models representam as entidades do domínio (Medicamento, Embalagem, Consumo, Preferências) e as suas relações.

Modelo de Dados

O modelo de dados foi desenhado para reflectir a realidade do domínio: um utilizador possui vários medicamentos no seu catálogo; cada medicamento pode ter várias embalagens com validades diferentes; cada embalagem pode registar múltiplos consumos ao longo do tempo.

Diagrama Entidade-Relacionamento:

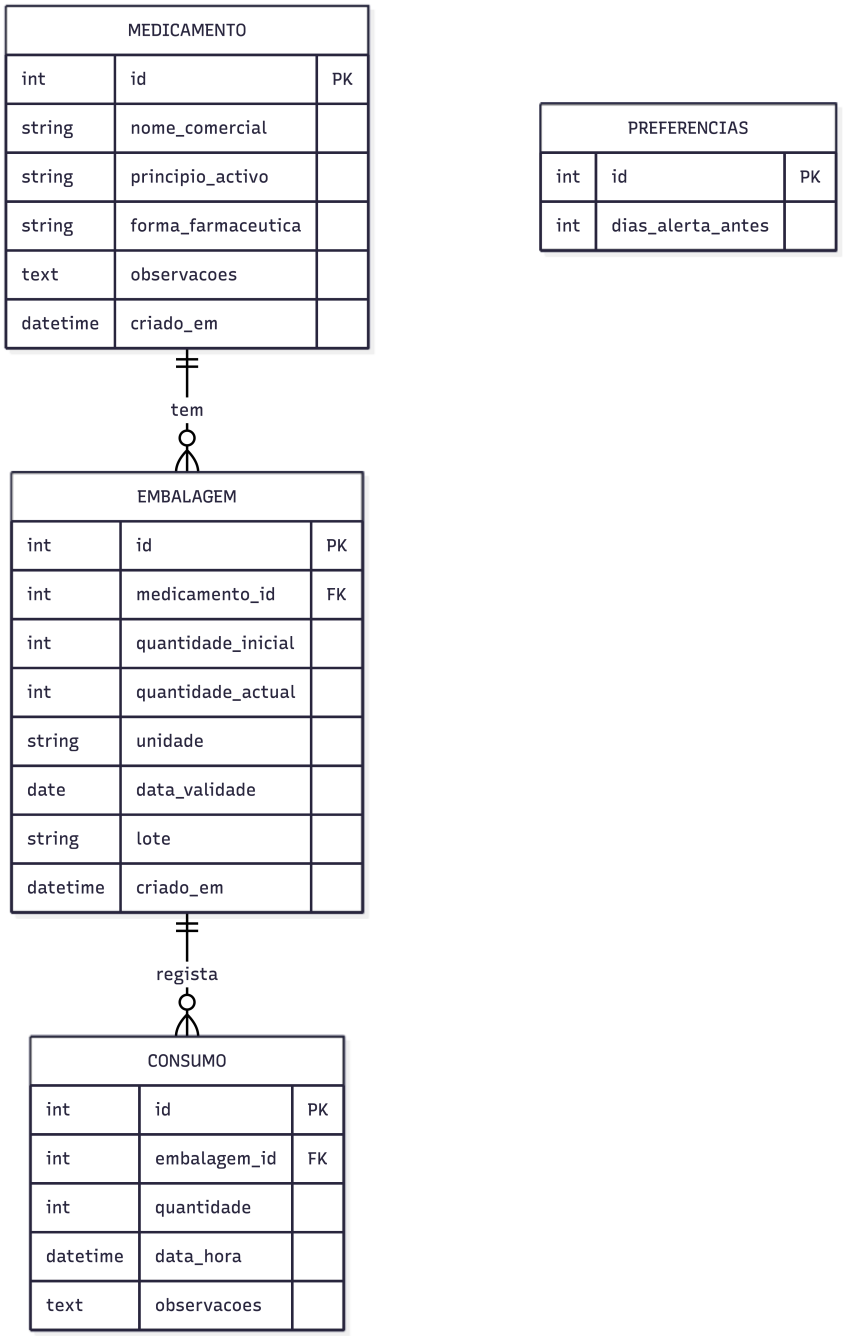

Figura 2 – Diagrama Relacional da Base de Dados

Entidades principais:

Entidade	Descrição
Medicamento	Catálogo de medicamentos. Armazena nome comercial, princípio activo, forma farmacêutica e observações.
Embalagem	Stock físico. Cada registo representa uma caixa/frasco com quantidade, unidade, data de validade e lote opcional.
Consumo	Registo de tomas/utilizações. Cada consumo desconta a quantidade da embalagem associada.
Preferências	Configurações do sistema. Define o número de dias de antecedência para os alertas de validade.

FUNCIONALIDADES DO MVP

O MVP (Minimum Viable Product) contempla as funcionalidades essenciais para demonstrar o valor da aplicação. Funcionalidades adicionais poderão ser desenvolvidas em fases posteriores.

F1 – Gestão de Medicamentos
Permite ao utilizador manter um catálogo dos medicamentos que possui ou já possuiu. O utilizador pode criar novos medicamentos, visualizar a lista completa, editar informações e eliminar registos. A eliminação de um medicamento remove também todas as embalagens e consumos associados (eliminação em cascata).

F2 – Gestão de Stock (Embalagens)
Permite registar o stock físico por embalagem. Cada embalagem está associada a um medicamento e possui quantidade inicial, quantidade actual, unidade de medida, data de validade e opcionalmente número de lote. A listagem de embalagens é ordenada por data de validade (mais antigas primeiro), seguindo o princípio FEFO (First Expired, First Out).

F3 – Registo de Consumos
Permite registar quando o utilizador toma ou utiliza um medicamento. O registo de consumo especifica a embalagem, a quantidade consumida e a data/hora. Ao guardar, a quantidade é automaticamente descontada do stock da embalagem. O sistema impede registar consumos superiores à quantidade disponível.

F4 – Sistema de Alertas de Validade
O sistema calcula e apresenta alertas sobre medicamentos expirados ou prestes a expirar. O utilizador pode configurar o número de dias de antecedência para os alertas (ex.: alertar 30 dias antes). Os alertas são apresentados de duas formas: num ícone de notificação (sino) na barra de navegação com badge indicando o número de alertas, e numa página dedicada que lista todas as embalagens em situação de alerta.

F5 – Dashboard
Página inicial que apresenta uma visão geral do estado da farmácia: alertas activos (expirados e a expirar), resumo do stock, e acções rápidas para as operações mais comuns (adicionar medicamento, registar consumo). O dashboard é o ponto de entrada principal da aplicação.

F6 – Preferências
Permite ao utilizador configurar parâmetros do sistema, nomeadamente o número de dias de antecedência para os alertas de validade. Esta configuração é persistida na base de dados e aplicada em todos os cálculos de alertas.

INTERFACE DO UTILIZADOR

Princípios de Design

A interface segue uma abordagem *mobile-first*, garantindo que a experiência em dispositivos móveis seja prioritária. O design é minimalista, focado na usabilidade e na clareza da informação apresentada.

Estrutura de Navegação

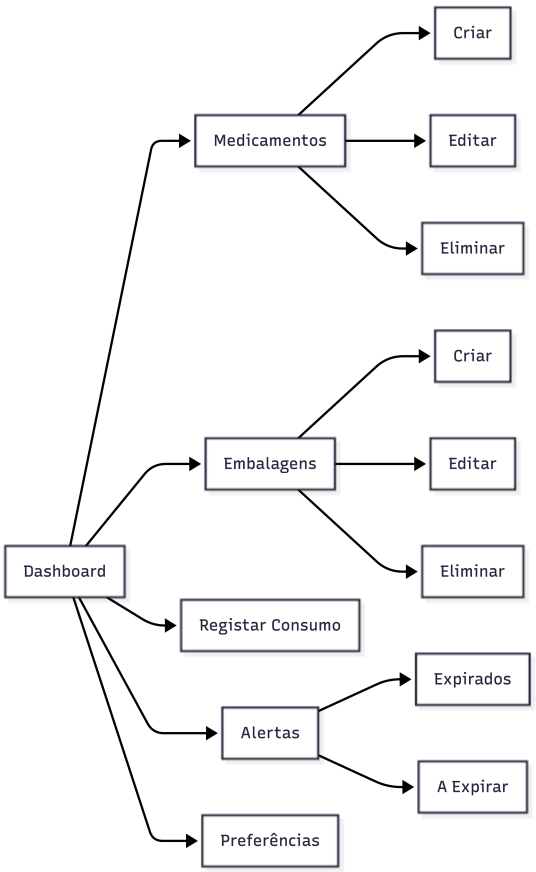

Figura 3 – Diagrama de Navegação

Páginas da Aplicação

Página	URL	Função
Dashboard	/	Visão geral, alertas, acções rápidas
Medicamentos	/medicamentos/	Lista de medicamentos no catálogo
Novo Medicamento	/medicamentos/novo/	Formulário de criação
Embalagens	/embalagens/	Lista de stock ordenada por validade
Nova Embalagem	/embalagens/nova/	Formulário de adição ao stock
Registar Consumo	/consumo/novo/	Formulário de registo de toma
Alertas	/alertas/	Lista de embalagens expiradas e a expirar
Preferências	/preferencias/	Configuração de dias de alerta

PLANO DE IMPLEMENTAÇÃO

Milestones

O desenvolvimento será organizado em milestones incrementais, permitindo validar o progresso e ajustar o plano conforme necessário.

Milestone	Descrição	Entregáveis
M1	Fundação: setup do projecto, models, migrações, admin	Projecto Django funcional
M2	Template base com Bootstrap, navbar responsiva	Interface visual base
M3	CRUD completo de Medicamentos	Gestão de catálogo
M4	CRUD de Embalagens, ordenação FEFO	Gestão de stock
M5	Registo de consumos com desconto automático	Tracking de tomas
M6	Dashboard, alertas, preferências	Sistema de notificações
M7	Polimento, fixtures, README, testes de entrega	Pacote final

Riscos e Mitigações

Risco	Mitigação
Scope creep	MVP estritamente definido; funcionalidades extra em backlog separado
Erros de modelo de dados	Validar modelos antes de implementar views; usar migrações Django
Incompatibilidade de ambiente	Testar instalação do zero seguindo o README antes de entregar
Gestão de datas/timezones	Usar DateField para validades; configurar TIME_ZONE em settings.py

ESTRATÉGIA DE ENTREGA

Repositório Git

O código-fonte será mantido num repositório Git privado no GitHub, com o professor convidado como colaborador. O histórico de commits reflectirá a evolução do desenvolvimento, permitindo verificar o progresso ao longo do tempo.

Estrutura do Repositório

O repositório conterá o código-fonte Django, ficheiros de configuração, README com instruções de instalação e execução, documentação (este guião e manual de utilizador), e opcionalmente fixtures com dados de exemplo.

Instruções de Execução

O README incluirá instruções passo-a-passo para executar a aplicação: criação de ambiente virtual Python, instalação de dependências via requirements.txt, execução de migrações, e arranque do servidor de desenvolvimento. Estas instruções serão testadas numa máquina limpa antes da entrega.

NOTA SOBRE UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL

No planeamento e documentação deste Guião foi utilizado o Claude AI da Anthropic como ferramenta de apoio. O Claude auxiliou nas seguintes áreas:

- Sugestões de arquitectura e padrões de design
- Revisão e estruturação da documentação
- Esclarecimento de conceitos técnicos

A utilização desta ferramenta permitiu acelerar o processo de aprendizagem e desenvolvimento, mantendo sempre o controlo e compreensão por parte do aluno sobre todas as decisões técnicas e conceptuais. A ideia original, bem como a escolhar das tecnologias e linguagens de programação parte da solução proposta, foram da inteira responsabilidade do aluno. O trabalho de implementação, testes e validação será integralmente realizado pelo aluno.

REFERÊNCIAS

DJANGO SOFTWARE FOUNDATION. *Django Documentation*. Disponível em: https://docs.djangoproject.com/
SQLITE CONSORTIUM. *SQLite Documentation*. Disponível em: https://www.sqlite.org/docs.html
BOOTSTRAP TEAM. *Bootstrap Documentation*. Disponível em: https://getbootstrap.com/docs/
GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley, 1994.
FOWLER, M. *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley, 2002.