

Projeto PSR 2025

Tiny Concerts Documentação Técnica

Documentação Técnica



Autores:

Miguel Ferreira – 41102

Tomás Oliveira – 2024119054

Curso: Engenharia Informática

Unidade Curricular: Projeto de Sistemas e Redes

Docente: Vítor Sousa

Data: 29 de Maio de 2025

Índice

Agradecimentos.....	2
Introdução.....	3
Contexto do Projeto.....	4
Objetivos do Projeto.....	4
Escopo do Projeto.....	5
Tecnologias Utilizadas.....	6
Arquitetura do Sistema.....	7
Requisitos do Sistema.....	8
Requisitos Funcionais:.....	8
Requisitos Não Funcionais:.....	9
Modelagem de Dados.....	10
Diagrama Entidade-Relacionamento (E-R).....	11
Diagrama de Casos de Uso.....	13
Diagrama de Sequência.....	15
Plano de Testes e Qualidade.....	17
Deploy e Infraestrutura.....	18
Segurança e Considerações Éticas.....	19
Capturas de Ecrã da plataforma.....	21
Conclusão e Trabalho Futuro.....	25
Bibliografia.....	27

Agradecimentos

Gostaríamos de expressar o nosso sincero agradecimento ao professor Vítor Sousa, pela sua orientação, apoio e disponibilidade ao longo de todo o desenvolvimento deste projeto. A sua experiência e dedicação foram fundamentais para o nosso aprendizado e sucesso.

Agradecemos também aos colegas de equipa, pela colaboração e empenho demonstrados durante todas as fases do trabalho.

Por fim, deixamos o nosso reconhecimento à Faculdade de Ciência e Tecnologia da Universidade Fernando Pessoa, pelo ambiente académico que promove a excelência e o desenvolvimento científico.

Este projeto só foi possível graças ao contributo e incentivo de todos, pelo que manifestamos a nossa profunda gratidão.

Introdução

Este relatório apresenta o desenvolvimento do projeto Tiny Concerts, uma plataforma web destinada ao streaming on-demand de concertos ao vivo, inspirada na série "Tiny Desk Concerts" da NPR Music. O principal objetivo do projeto foi criar uma aplicação que permitisse aos utilizadores explorar, assistir e interagir com vídeos de concertos, proporcionando uma experiência personalizada e rica em funcionalidades.

O desenvolvimento deste sistema foi realizado no âmbito da disciplina de Sistemas e Redes da Faculdade de Ciência e Tecnologia da Universidade Fernando Pessoa, com enfoque na integração de tecnologias modernas para o frontend, backend e base de dados, bem como na aplicação de boas práticas de engenharia de software, como testes automatizados, integração contínua e deployment em cloud.

Nesta introdução, são apresentados o contexto que motivou o projeto, os seus objetivos específicos e o âmbito das funcionalidades implementadas, oferecendo ao leitor uma visão geral do trabalho desenvolvido e da sua relevância.

Contexto do Projeto

Nos últimos anos, a forma como as pessoas consomem música tem vindo a mudar significativamente, com uma crescente preferência por plataformas digitais que permitem aceder a conteúdos multimédia de forma rápida e personalizada. A série "Tiny Desk Concerts" da NPR Music tornou-se um fenómeno ao disponibilizar concertos intimistas, filmados num ambiente descontraído, que rapidamente conquistaram milhares de fãs a nível mundial.

Este projeto surge da vontade de criar uma plataforma própria que permita a disponibilização destes concertos num formato acessível a qualquer utilizador, promovendo a interação social através de funcionalidades como comentários, likes e favoritos. A plataforma pretende, assim, proporcionar uma experiência mais rica e interativa, refletindo as tendências atuais do consumo digital de música.

Objetivos do Projeto

O principal objetivo do projeto Tiny Concerts foi desenvolver uma aplicação web funcional que permita aos utilizadores:

- Registrar-se e autenticar-se de forma segura na plataforma;
- Explorar e pesquisar concertos disponíveis, com filtros por artista, gênero musical e data;
- Assistir aos vídeos de concertos diretamente no browser, com um player incorporado;
- Interagir com os vídeos, através da possibilidade de os favoritar, gostar e comentar;
- Reportar problemas relacionados com os vídeos ou a plataforma;
- Gerir a plataforma através de um painel administrativo, que permite adicionar, editar e remover concertos;

Adicionalmente, o projeto procurou assegurar a implementação de medidas de segurança, garantir a qualidade do software com testes automatizados, e automatizar o processo de integração e deployment contínuo (CI/CD).

Escopo do Projeto

O escopo do projeto Tiny Concerts inclui o desenvolvimento de uma plataforma web para transmissão de concertos ao vivo, com funcionalidades essenciais que garantem uma experiência de utilizador completa e segura.

O sistema contempla as seguintes funcionalidades principais:

- Registo, autenticação e gestão de utilizadores, com diferentes níveis de acesso (utilizadores comuns e administradores);
- Pesquisa e filtragem de concertos por artista, género musical e data de lançamento;
- Reprodução de vídeos diretamente no navegador, com funcionalidades de interação como gostos, favoritos e comentários;
- Sistema para reportar problemas relacionados com os vídeos ou a plataforma;
- Painel administrativo para gerir os concertos, incluindo a adição, edição e remoção de conteúdos.
- O projeto focou-se na criação de um protótipo funcional (MVP) que assegura a integração eficiente entre o frontend, backend e base de dados, utilizando boas práticas de desenvolvimento, testes automatizados e deployment em ambiente cloud.

Funcionalidades avançadas, como sistemas de recomendação, gamificação e integração com APIs externas, foram deixadas fora do escopo inicial, podendo ser implementadas em fases futuras.

Tecnologias Utilizadas

Para o desenvolvimento da plataforma Tiny Concerts foram selecionadas tecnologias modernas e adequadas ao propósito do projeto, garantindo robustez, escalabilidade e facilidade de manutenção.

Frontend

O frontend foi desenvolvido em Angular, um framework popular para construção de aplicações web dinâmicas e responsivas. O Angular permitiu criar uma interface intuitiva e eficiente, compatível com diferentes dispositivos, garantindo uma boa experiência de utilizador tanto em desktop como em dispositivos móveis.

Backend

No backend, utilizou-se o Flask, um micro-framework em Python que facilita a criação de APIs RESTful leves e modulares. O Flask foi escolhido pela sua simplicidade, flexibilidade e pela facilidade de integração com bases de dados relacionais.

Base de Dados

A base de dados escolhida foi o PostgreSQL, um sistema de gestão de base de dados relacional avançado e de código aberto, reconhecido pela sua robustez, desempenho e suporte a funcionalidades avançadas, como integridade referencial e transações complexas.

Testes e Ferramentas de Desenvolvimento

Para garantir a qualidade e a manutenção do código, foram utilizadas diversas ferramentas e frameworks, entre as quais:

SQLAlchemy: ORM para facilitar a manipulação da base de dados a partir do backend Python.

pytest e unittest: frameworks para testes automatizados no backend.

Karma e Jasmine: frameworks para testes no frontend Angular.

Git e GitHub: para controlo de versões e colaboração no desenvolvimento do código-fonte.

GitHub Actions: para automação da integração contínua (CI) e deployment contínuo (CD).

Docker: para criação de ambientes isolados, garantindo consistência entre desenvolvimento e produção.

Deploy

O deployment do projeto foi realizado através da plataforma Render, que permite a automatização do processo de disponibilização do backend e frontend na cloud, garantindo escalabilidade, fiabilidade e facilidade na gestão da infraestrutura.

Arquitetura do Sistema

A arquitetura da plataforma Tiny Concerts foi concebida para garantir modularidade, escalabilidade e manutenção facilitada, adotando uma separação clara entre as componentes de frontend, backend e base de dados.

Estrutura Geral

O sistema é organizado em três camadas principais: Frontend, Backend e Base de Dados.

Frontend

Desenvolvido em Angular, o frontend é responsável pela interface do utilizador, permitindo a interação com a plataforma através de páginas web responsivas. Comunica com o backend via APIs REST, consumindo dados e enviando pedidos de autenticação, pesquisa, interação com vídeos e comentários.

Backend

Implementado em Flask, o backend fornece uma API RESTful que serve de intermediário entre o frontend e a base de dados. É responsável pela lógica de negócio, autenticação, validação de dados, controlo de acessos (RBAC) e gestão dos dados armazenados.

Base de Dados

Utiliza PostgreSQL para armazenamento relacional dos dados da aplicação, incluindo utilizadores, concertos, interações (likes, favoritos, comentários) e relatórios. A estrutura foi normalizada para garantir integridade e eficiência nas operações.

Comunicação e Fluxo de Dados

O frontend comunica com o backend através de chamadas HTTP (REST API), trocando dados em formato JSON. O backend processa as requisições, executa as operações necessárias na base de dados e devolve respostas estruturadas para o frontend.

Requisitos do Sistema

Requisitos Funcionais:

Os requisitos funcionais definem as funcionalidades que o sistema deve assegurar para cumprir os objetivos do projeto Tiny Concerts:

1. Registo e Autenticação de Utilizadores

- a. Permitir o registo de novos utilizadores com nome, email e password.
- b. Validar a unicidade do email durante o registo.
- c. Permitir o login e logout seguro dos utilizadores.
- d. Implementar controlo de acesso baseado em papéis (utilizador comum e utilizador administrador).

2. Gestão de Perfil do Utilizador

- a. Permitir a visualização e edição dos dados pessoais do utilizador.
- b. Exibir listas de vídeos favoritos e vídeos com “like” do utilizador.

3. Navegação e Pesquisa de Concertos

- a. Exibir lista de concertos com título, artista e miniatura.
- b. Permitir pesquisa por título, artista, género e data.
- c. Permitir aplicação de filtros para facilitar a busca.

4. Visualização de Vídeos de Concertos

- a. Reproduzir vídeos incorporados diretamente na página.
- b. Permitir controlar a reprodução (play, pause, volume).
- c. Permitir aos utilizadores marcar vídeos como favoritos e dar “like”.
- d. Possibilitar a inserção e visualização de comentários em cada vídeo.
- e. Permitir aos utilizadores reportar problemas associados a vídeos ou à plataforma.

5. Painel Administrativo

- a. Permitir a criação, edição e remoção de concertos pela equipa administrativa.
- b. Gerir os dados dos concertos e assegurar a integridade da informação.

Requisitos Não Funcionais:

Os requisitos não funcionais definem características qualitativas do sistema, como desempenho, segurança e usabilidade:

1. Segurança

- a. Utilizar encriptação para proteger dados sensíveis, nomeadamente passwords.
- b. Garantir comunicações seguras via HTTPS.
- c. Assegurar que funcionalidades administrativas são restritas a utilizadores autorizados.

2. Desempenho

- a. O sistema deve responder às requisições do utilizador num tempo aceitável, garantindo fluidez na navegação.
- b. A plataforma deve ser capaz de suportar múltiplos utilizadores simultâneos.

3. Usabilidade

- a. A interface deve ser intuitiva e responsiva, compatível com dispositivos móveis e desktop.
- b. Os processos de registo, pesquisa e visualização devem ser simples e diretos.

4. Manutenibilidade

- a. O código deve ser modular e bem documentado para facilitar futuras atualizações.
- b. Devem ser implementados testes automatizados para garantir a qualidade do software.

5. Disponibilidade e Escalabilidade

- a. O sistema deve estar disponível 24/7, com mínimas interrupções.
- b. Deve ser possível escalar a aplicação conforme o número de utilizadores aumentar.

Modelagem de Dados

A modelagem de dados teve como objetivo estruturar de forma clara e eficiente a informação que a plataforma Tiny Concerts irá armazenar e gerir. Para isso, foram elaborados dois diagramas fundamentais:

Diagrama Entidade-Relacionamento (E-R):

Diagrama que identifica as principais entidades do sistema, os seus atributos e os relacionamentos entre elas, garantindo a integridade e a coerência dos dados. Este diagrama representa graficamente a estrutura da base de dados e facilita o seu desenvolvimento.

Diagrama de Casos de Uso:

Diagrama que descreve as interações entre os diferentes atores do sistema e as funcionalidades disponibilizadas. Este diagrama permite esclarecer os requisitos funcionais, mostrando como utilizadores e administradores interagem com a plataforma.

Diagrama de Sequência:

Diagrama que representa a interação entre os diferentes objectos do sistema ao longo do tempo, evidenciando a ordem das mensagens trocadas. Este diagrama ilustra, de forma temporal, como os processos são executados, demonstrando a lógica de funcionamento de cada funcionalidade.

Estes três diagramas — o Diagrama Entidade-Relacionamento, o Diagrama de Casos de Uso e o Diagrama de Sequência — foram fundamentais para garantir que a estrutura dos dados, os requisitos funcionais e o comportamento do sistema ao longo do tempo estão bem definidos. Em conjunto, asseguraram um desenvolvimento mais eficiente, coerente e alinhado com os objectivos do projecto.

Diagrama Entidade-Relacionamento (E-R)

O **diagrama E-R** representa graficamente as entidades do sistema e as suas relações, permitindo visualizar a estrutura da base de dados e facilitar o seu desenvolvimento. As principais entidades modeladas foram:

Users: Representa os utilizadores da plataforma, contendo atributos como identificação, nome, email, password, tipo de utilizador e informações de controlo (datas de criação, atualização, etc.).

UserType: Define os tipos de utilizadores (por exemplo, administrador ou utilizador comum), permitindo implementar controlos de acesso.

Concerts: Representa os concertos disponíveis na plataforma, com atributos como título, artista, URL do vídeo, género e data de lançamento.

Likes e Favorites: Entidades associativas que registam as interações dos utilizadores com os concertos, permitindo que um utilizador possa gostar e guardar concertos como favoritos.

Comments: Representa os comentários feitos pelos utilizadores em cada concerto, armazenando texto e data de criação.

Reports: Regista os relatórios de problemas feitos pelos utilizadores relativamente a concertos, incluindo o tipo de problema e descrição.

Diagrama Entidade-Relacionamento (E-R)

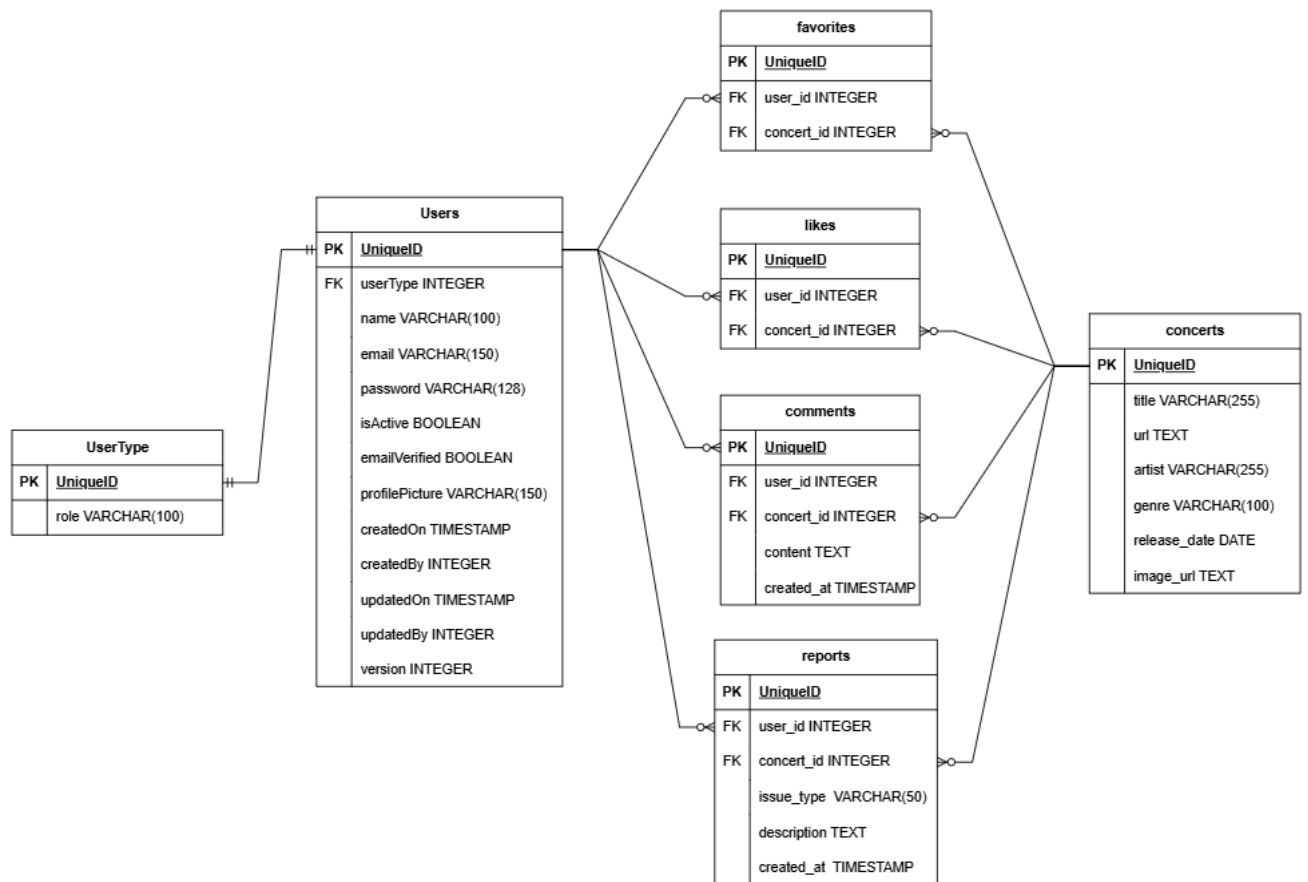


Diagrama de Casos de Uso

O **diagrama de casos de uso** foi desenvolvido para ilustrar as interações entre os diferentes atores do sistema e as funcionalidades disponíveis na plataforma Tiny Concerts. Este diagrama permite compreender de forma clara os requisitos funcionais e as responsabilidades de cada tipo de utilizador, facilitando o planeamento e a implementação do sistema.

Atores Principais

Utilizador:

Este ator representa o utilizador final da plataforma, que pode registar-se, iniciar sessão, gerir o seu perfil e interagir com os conteúdos disponíveis.

As principais ações do utilizador incluem:

Registar conta, Iniciar sessão, Editar perfil, Pesquisar concertos, Visualizar lista de concertos, Ver vídeo do concerto, Gostar de vídeo, Adicionar vídeo aos favoritos, Comentar vídeo e Reportar problema no vídeo

Administrador:

O administrador para além de possuir todas as funcionalidades do utilizador possui privilégios adicionais para gerir os conteúdos da plataforma, assegurando a sua atualização e qualidade.

As funcionalidades extra atribuídas ao administrador são:

Iniciar sessão como administrador, Adicionar concerto, Editar concerto e Remover concerto

Sistema:

O sistema representa funcionalidades automáticas que suportam a operação da plataforma, tais como:

Validar email durante o registo, Autenticar utilizadores e administradores, Filtrar resultados da pesquisa e Guardar interações (gostos, favoritos, comentários)

Diagrama de Casos de Uso

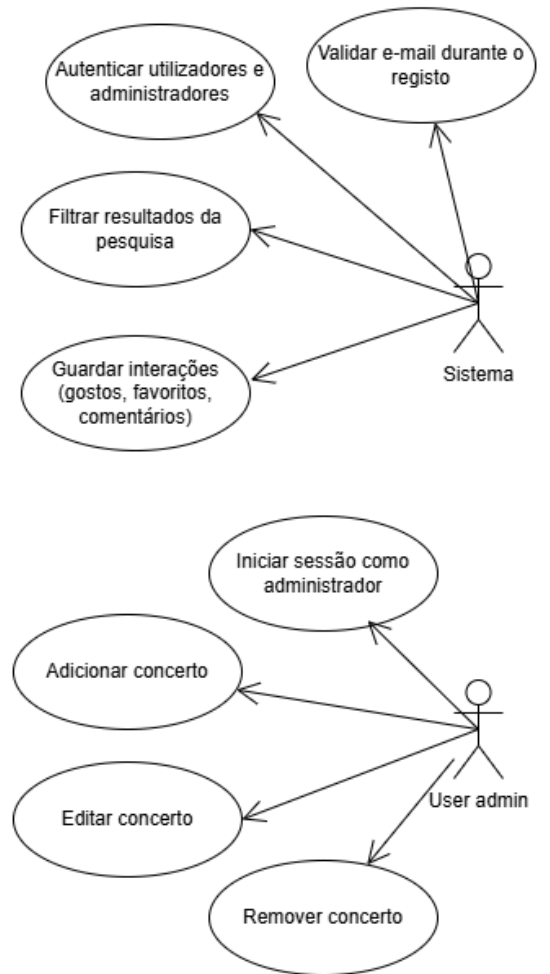
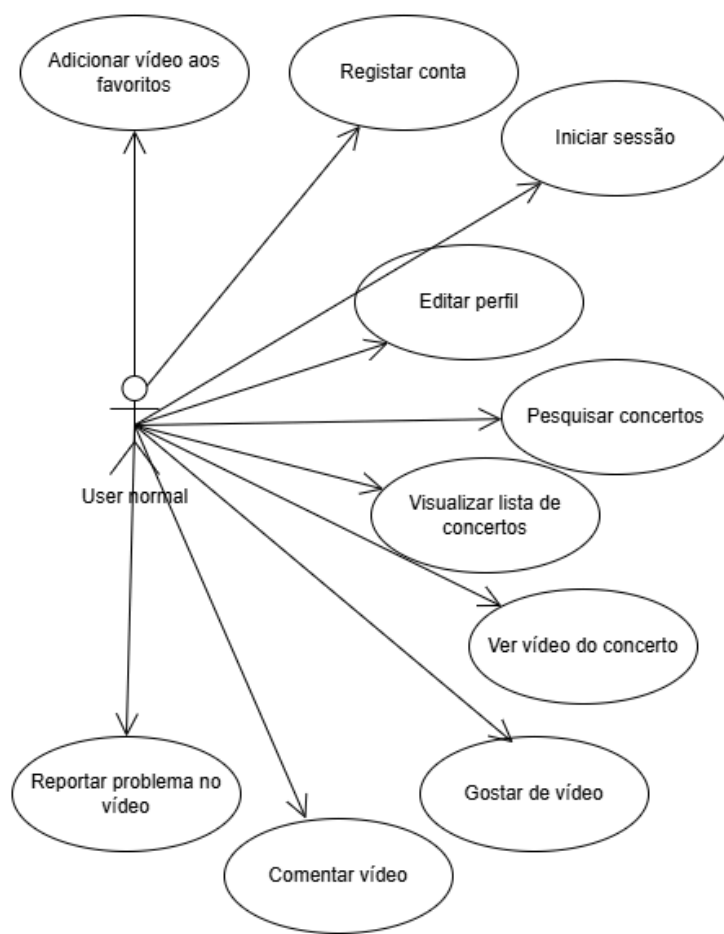


Diagrama de Sequência

O **diagrama de sequência** apresentado descreve, de forma detalhada, a interação entre os principais componentes envolvidos na visualização do detalhe de um concerto na plataforma Tiny Concerts. Este tipo de diagrama permite compreender a ordem das mensagens trocadas entre frontend, backend e base de dados, revelando como o sistema responde às ações do utilizador em tempo real.

Componentes envolvidos

Utilizador: Interage com a interface para visualizar o concerto e realizar ações como comentar, gostar ou adicionar aos favoritos.

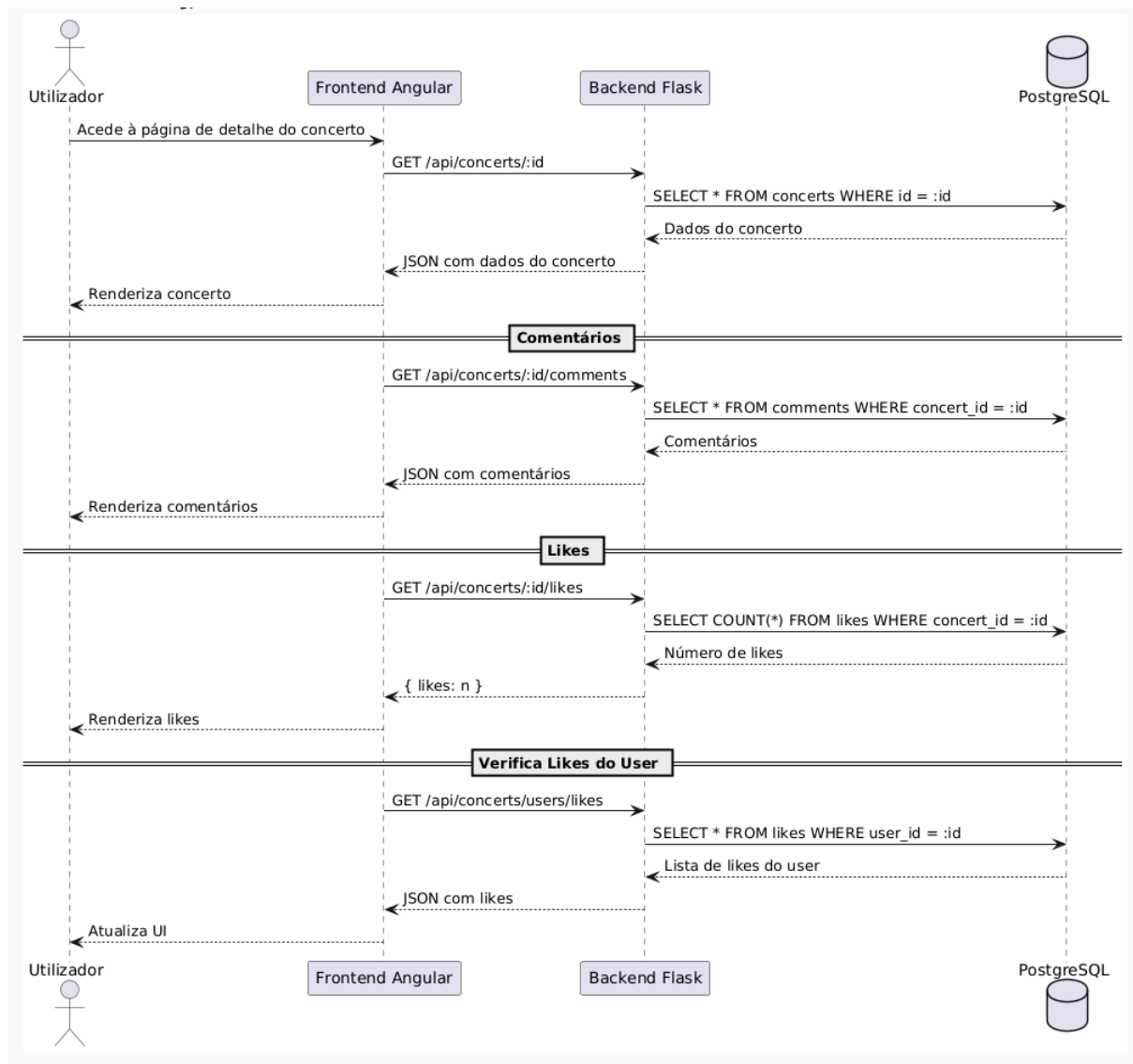
Frontend (Angular): Responsável por fazer as chamadas HTTP para o backend e apresentar os dados ao utilizador.

Backend (Flask API): Recebe os pedidos do frontend, processa a lógica de negócio e interage com a base de dados.

Base de Dados (PostgreSQL): Armazena toda a informação relacionada com concertos, utilizadores, comentários, likes e favoritos.

Importância: Este diagrama é essencial para compreender o comportamento dinâmico da aplicação, garantir a correta comunicação entre os componentes e ajudar na deteção de erros, como falhas na autenticação ou ausência de dados esperados. Também serve como referência técnica para manutenção e futuras expansões do sistema.

Diagrama de Sequência



Plano de Testes e Qualidade

Para garantir a robustez, fiabilidade e manutenção do sistema Tiny Concerts, foi implementado um plano abrangente de testes e controlo de qualidade durante todas as fases do desenvolvimento. Este plano contempla várias metodologias e ferramentas para assegurar que o software cumpre os requisitos funcionais e não funcionais definidos.

Testes Automatizados

Backend:

O backend, desenvolvido em Flask, foi sujeito a testes automatizados utilizando os frameworks pytest e unittest. Estes testes focaram-se em validar as funcionalidades da API, incluindo registo e autenticação de utilizadores, gestão de concertos, e operações relacionadas com likes, favoritos e comentários. A cobertura de testes visa garantir que os principais fluxos de dados e casos de erro estão devidamente tratados.

Frontend:

No frontend Angular, foram implementados testes unitários e de integração utilizando as ferramentas Karma e Jasmine. Estes testes asseguram que os componentes da interface respondem corretamente às interações do utilizador e que a comunicação com o backend ocorre conforme esperado.

Análise de Qualidade de Código

Para manter a qualidade do código e evitar erros comuns, foram utilizadas ferramentas de análise estática, como:

ESLint no frontend para detetar problemas de sintaxe, estilo e potenciais bugs em JavaScript/TypeScript.

Flake8 e Pylint no backend para garantir a conformidade com boas práticas de programação em Python.

Integração Contínua (CI) e Deployment Contínuo (CD)

Foi configurado um pipeline de integração contínua usando GitHub Actions, que automatiza:

A execução dos testes a cada push no repositório, garantindo que novas alterações não introduzam erros.

A análise automática da qualidade do código.

O deployment automatizado na plataforma Render, assegurando que a aplicação está sempre atualizada em ambiente de produção.

Resultados e Melhorias:

Graças a este plano de testes e controlo de qualidade, o sistema alcançou um nível elevado de estabilidade e segurança. Foram identificados e corrigidos bugs precocemente, e o desenvolvimento tornou-se mais eficiente e organizado. O processo permitiu também preparar o projeto para futuras expansões e manutenção simplificada.

Deploy e Infraestrutura

Para garantir a disponibilização eficiente, escalável e segura da plataforma Tiny Concerts, o deployment do sistema foi realizado na plataforma cloud Render. Esta solução permite gerir de forma integrada o frontend, backend e a base de dados, simplificando a configuração e manutenção da infraestrutura.

Deploy do Frontend:

O frontend Angular foi implantado na Render, aproveitando a capacidade da plataforma para servir aplicações estáticas de forma rápida e fiável. O processo de deployment foi automatizado através do pipeline de integração contínua, garantindo que as alterações no código-fonte são refletidas imediatamente na aplicação em produção.

Deploy do Backend:

O backend desenvolvido em Flask foi também implantado na Render, beneficiando da sua flexibilidade para hospedar aplicações Python com suporte a bases de dados PostgreSQL. O deploy automatizado assegura a disponibilidade constante da API, com monitorização integrada e escalabilidade conforme a procura dos utilizadores.

Base de Dados:

A base de dados PostgreSQL utilizada pelo sistema está alojada na Render, aproveitando os serviços geridos da plataforma para garantir a persistência, segurança e desempenho dos dados. A infraestrutura suporta backups automáticos e recuperação de dados, minimizando riscos de perda de informação.

Integração Contínua e Automatização:

O processo de integração e deployment contínuos foi implementado usando GitHub Actions, que, ao combinar-se com a Render, permite que cada alteração no repositório dispare automaticamente a construção, testes e publicação da aplicação. Este fluxo garante que a versão em produção esteja sempre atualizada, minimizando erros e tempo de inatividade.

Segurança e Considerações Éticas

A segurança dos dados e a proteção da privacidade dos utilizadores foram prioridades essenciais no desenvolvimento da plataforma Tiny Concerts. Foram implementadas diversas medidas para assegurar a integridade, confidencialidade e disponibilidade dos dados, bem como para garantir um uso responsável da informação.

Segurança

Autenticação e Autorização:

O sistema utiliza JSON Web Tokens (JWT) para autenticar utilizadores, assegurando sessões seguras e sem estado. O controlo de acesso é baseado em papéis (RBAC), garantindo que apenas utilizadores autorizados podem aceder a funcionalidades administrativas, enquanto utilizadores comuns têm permissões limitadas.

Encriptação de Dados Sensíveis:

As passwords dos utilizadores são armazenadas de forma segura, utilizando técnicas de hashing com algoritmos robustos, para prevenir acesso indevido mesmo em caso de violação da base de dados.

Comunicações Seguras:

A plataforma suporta HTTPS para garantir que todas as comunicações entre cliente e servidor são encriptadas, prevenindo ataques de interceptação e manipulação de dados.

Validação e Sanitização de Dados:

Todos os dados fornecidos pelos utilizadores são validados e sanitizados para evitar vulnerabilidades comuns, como injeção SQL e ataques de Cross-Site Scripting (XSS).

Segurança e Considerações Éticas

Considerações Éticas

Privacidade dos Utilizadores:

Os dados pessoais recolhidos são utilizados exclusivamente para os fins da plataforma, respeitando os princípios de minimização de dados e finalidade explícita. Não são partilhados com terceiros sem consentimento prévio.

Transparência e Consentimento:

A plataforma informa claramente os utilizadores sobre a recolha e uso dos seus dados, promovendo um ambiente de confiança e conformidade com regulamentos de proteção de dados.

Responsabilidade na Moderação:

O sistema permite reportar conteúdos inadequados, e a equipa administrativa tem a responsabilidade de analisar e agir conforme as políticas definidas, garantindo um ambiente seguro e respeitador para todos os utilizadores.

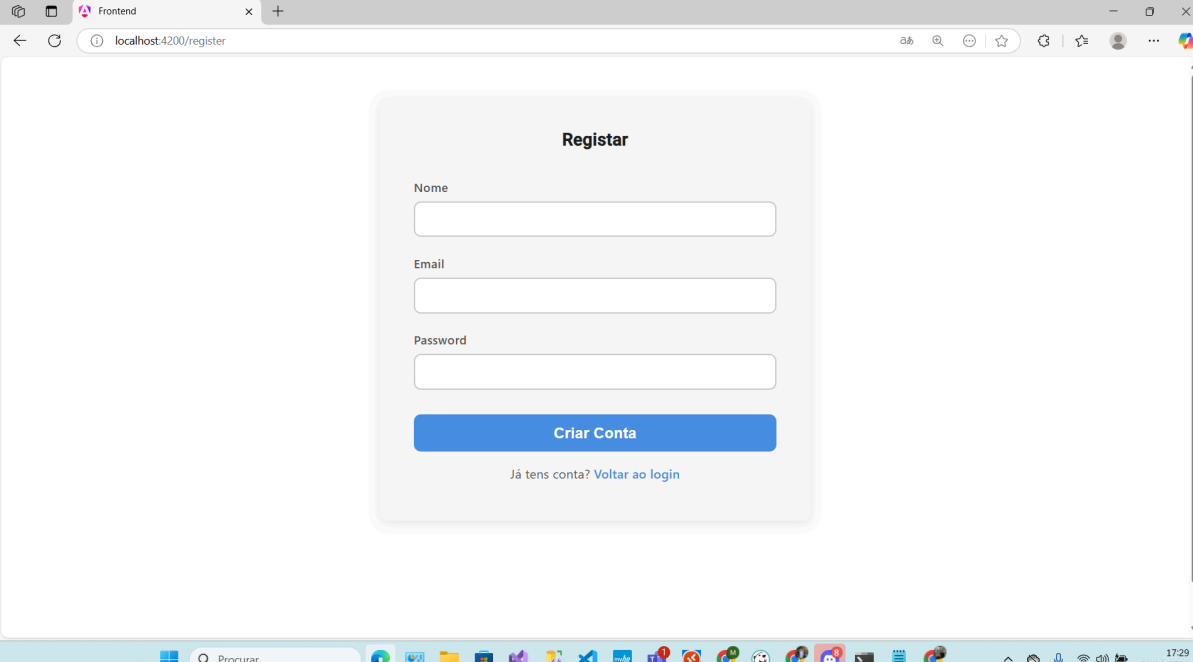
Acessibilidade e Inclusão:

Foram adotadas boas práticas para tornar a plataforma acessível a um público diversificado, promovendo a inclusão digital.

Estas medidas combinadas contribuem para a construção de uma plataforma segura, ética e alinhada com as melhores práticas em desenvolvimento de software e proteção de dados.

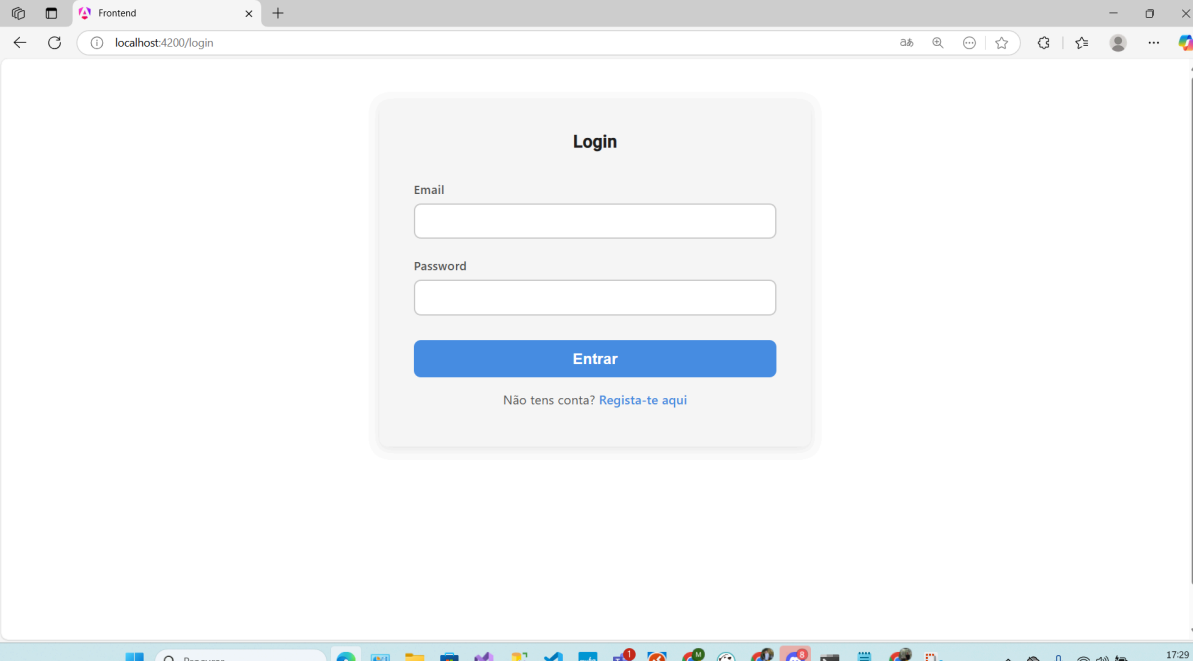
Capturas de Ecrã da plataforma

Página Registo:



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/register'. The page features a light gray background with a central white rounded rectangle containing the registration form. The form is titled 'Registrar' in bold black text. It includes three input fields: 'Nome' (Name), 'Email', and 'Password', each with a placeholder text. Below the fields is a blue button labeled 'Criar Conta'. At the bottom of the form, there is a link that says 'Já tens conta? Voltar ao login'.

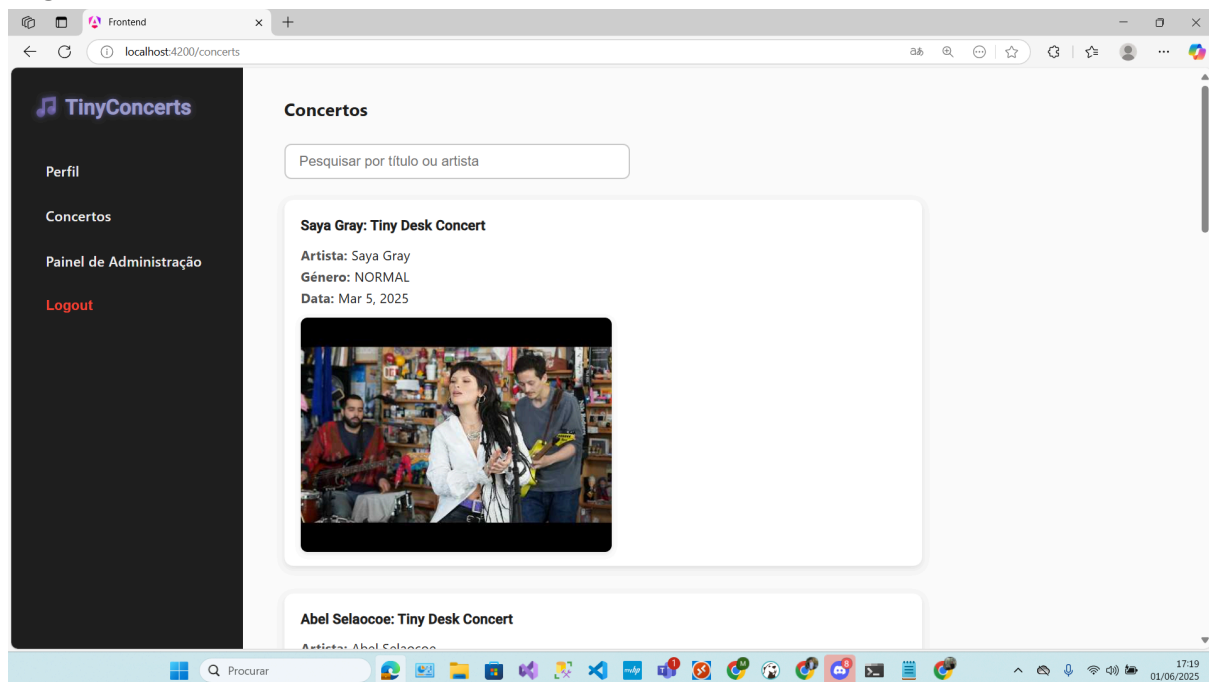
Página Login:



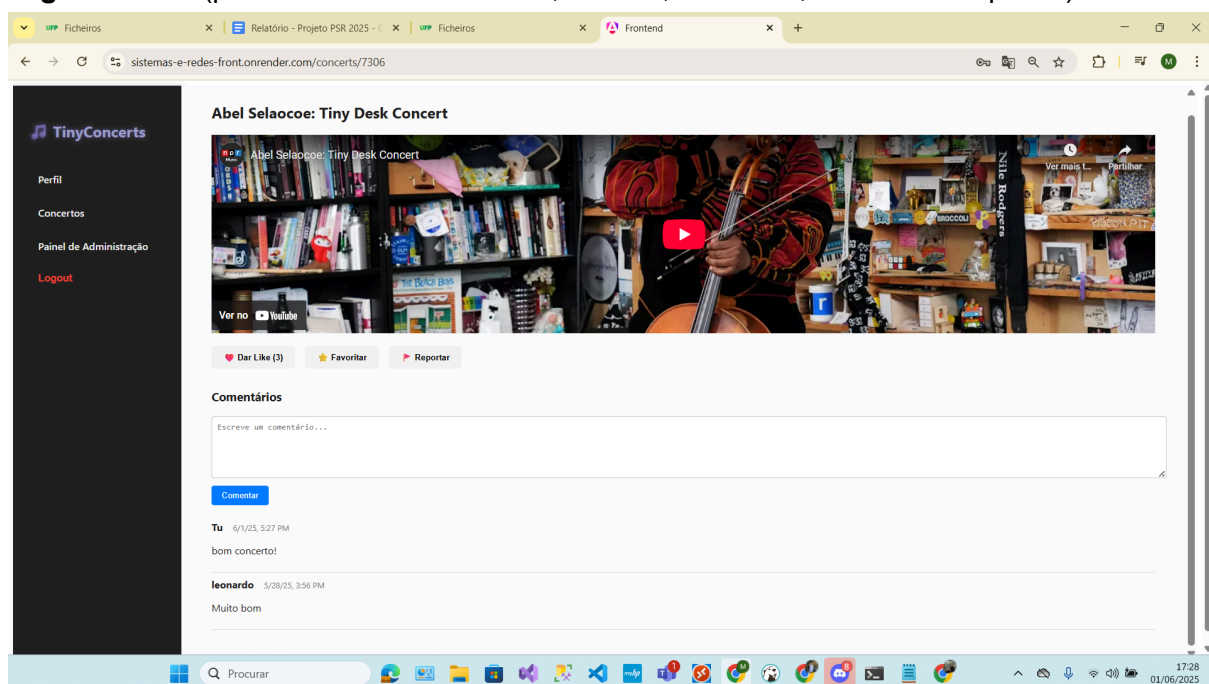
The screenshot shows a web browser window with the address bar displaying 'localhost:4200/login'. The page features a light gray background with a central white rounded rectangle containing the login form. The form is titled 'Login' in bold black text. It includes two input fields: 'Email' and 'Password', each with a placeholder text. Below the fields is a blue button labeled 'Entrar'. At the bottom of the form, there is a link that says 'Não tens conta? Regista-te aqui'.

Capturas de Ecrã da plataforma

Página Dashboard/Feed:

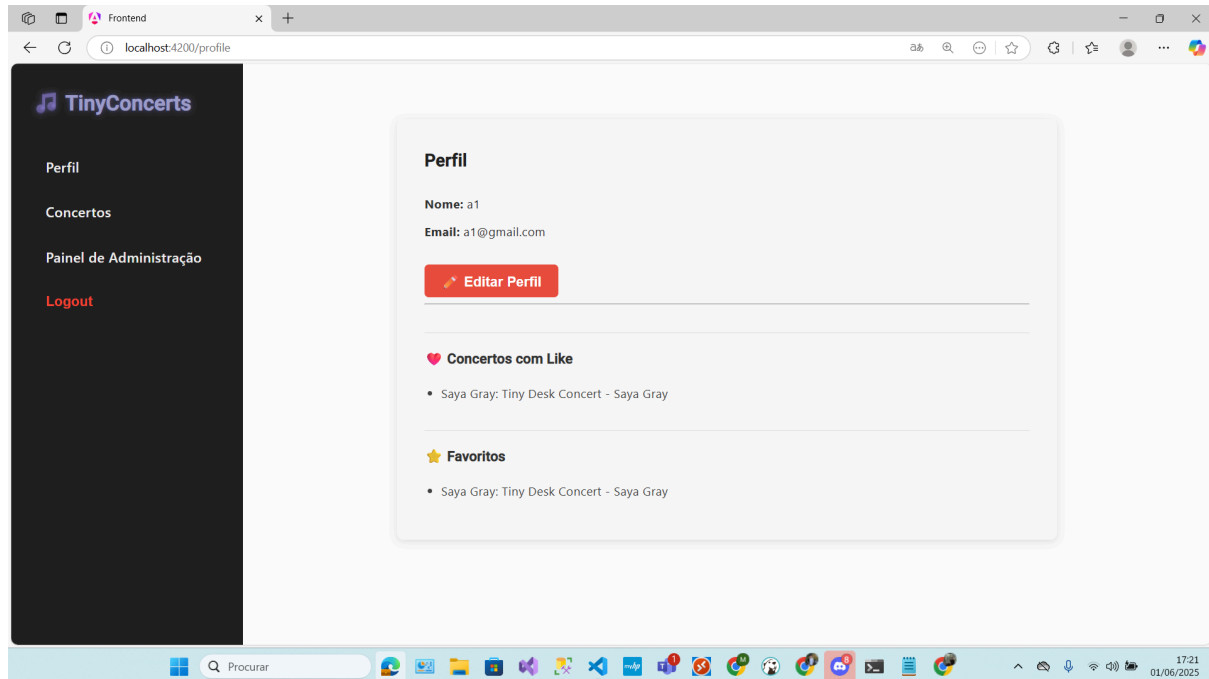


Página vídeo: (possibilidade de ver vídeo, dar like, comentar, favoritar e reportar)

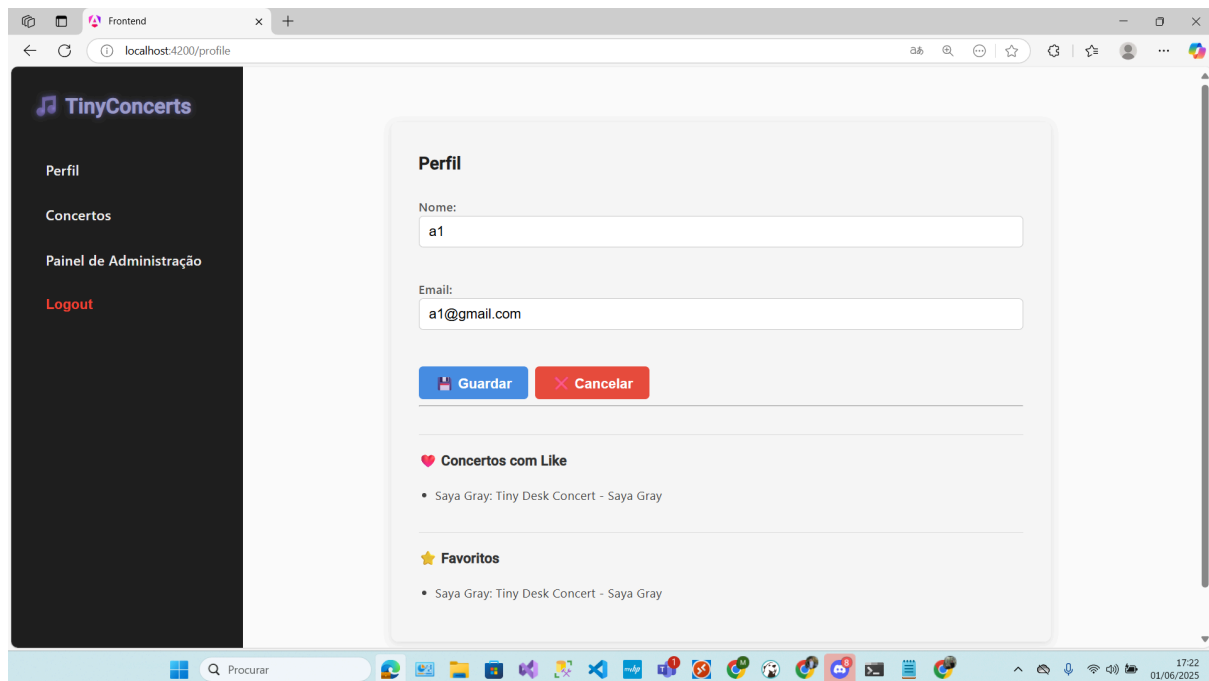


Capturas de Ecrã da plataforma

Página Perfil:

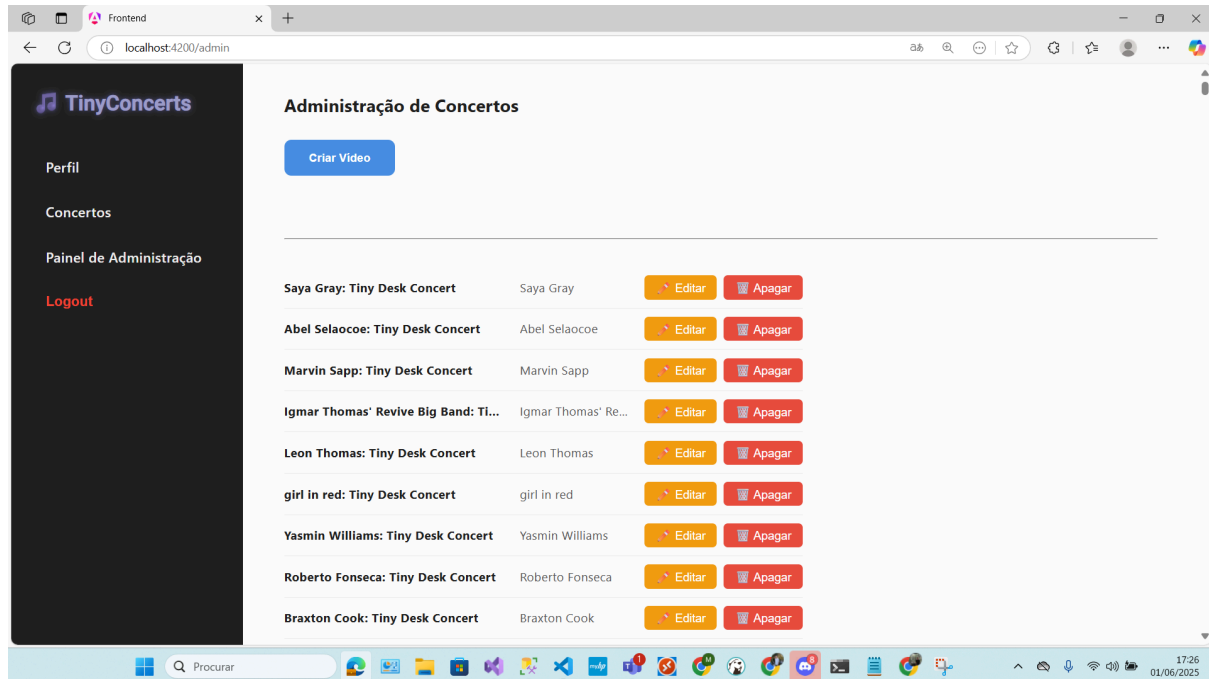


Página Perfil 2:

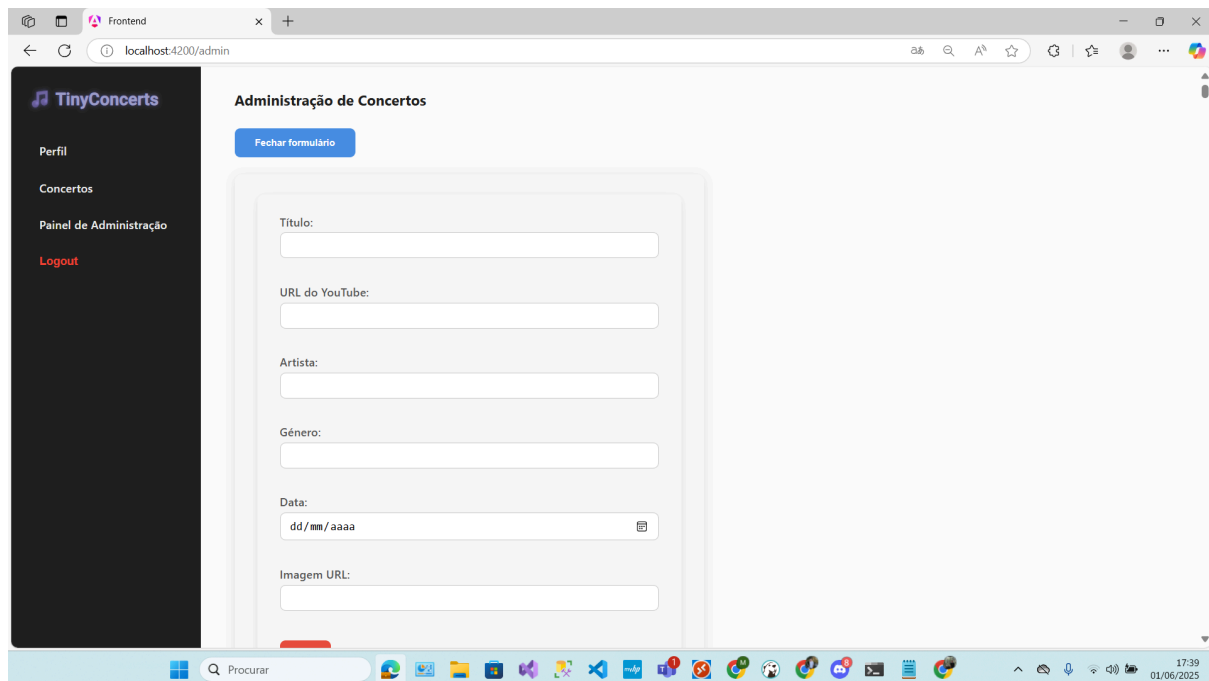


Capturas de Ecrã da plataforma

Página Painel de Administração:



Página Painel de Administração 2:



Conclusão e Trabalho Futuro

Conclusão

O projeto Tiny Concerts permitiu desenvolver uma plataforma web funcional e integrada para streaming on-demand de concertos ao vivo, cumprindo os principais objetivos definidos no início do trabalho. Através da combinação de tecnologias modernas como Angular, Flask e PostgreSQL, foi possível criar uma solução escalável, segura e com uma boa experiência de utilizador.

Durante o desenvolvimento, implementámos funcionalidades essenciais como registo e autenticação de utilizadores, pesquisa e visualização de concertos, interação social através de likes, favoritos e comentários, bem como um painel administrativo para gestão dos conteúdos. O uso de boas práticas de engenharia de software, incluindo testes automatizados, análise de qualidade de código e integração contínua, garantiu a robustez e a manutenibilidade do sistema.

O deployment automatizado na plataforma Render assegurou a disponibilização da aplicação em ambiente cloud, facilitando a gestão e escalabilidade da infraestrutura.

Conclusão e Trabalho Futuro

Trabalho Futuro / Futuras Melhorias

Apesar do sucesso alcançado, existem várias melhorias e funcionalidades adicionais que poderão ser implementadas para enriquecer a plataforma:

Progressive Web Application (PWA):

Otimizar a plataforma para oferecer uma experiência mobile nativa, permitindo utilização offline e notificações push.

Sistema de Recomendação:

Implementar algoritmos que sugiram concertos personalizados aos utilizadores com base nas suas preferências e histórico de visualização.

Gamificação:

Introduzir mecanismos de pontos, badges e reconhecimento comunitário para incentivar a interação dos utilizadores.

Integração com APIs Externas:

Adicionar serviços que forneçam informação adicional, como datas de concertos ao vivo, biografias de artistas e notícias relacionadas.

Importação Automática de Dados:

Permitir que os administradores façam upload de ficheiros CSV para atualização massiva dos concertos, facilitando a gestão do catálogo.

Melhorias na Acessibilidade:

Continuar a aprimorar a interface para garantir que a plataforma seja acessível a todos os utilizadores, incluindo aqueles com necessidades especiais.

Estas futuras melhorias permitirão ampliar o alcance e a qualidade da plataforma, consolidando o Tiny Concerts como uma solução completa e inovadora para o consumo digital de concertos.

Bibliografia

Durante o desenvolvimento do projeto Tiny Concerts, foram consultadas diversas fontes e recursos que apoiaram a compreensão e aplicação das tecnologias utilizadas. Destacamos as seguintes referências:

Angular

Documentação oficial do Angular, disponível em:

<https://angular.io/docs>

[Acesso em: Maio 2025]

Flask

Documentação oficial do Flask, framework Python para desenvolvimento web:

<https://flask.palletsprojects.com/en/latest/>

[Acesso em: Maio 2025]

PostgreSQL

Documentação oficial do sistema de gestão de base de dados PostgreSQL:

<https://www.postgresql.org/docs/>

[Acesso em: Maio 2025]

Render

Plataforma para deployment de aplicações web, utilizada para hospedar frontend, backend e base de dados:

<https://render.com/docs>

[Acesso em: Maio 2025]

GitHub Actions

Serviço de integração contínua e deployment contínuo (CI/CD) utilizado para automatizar build, testes e deploy:

<https://docs.github.com/en/actions>

[Acesso em: Maio 2025]

Material das Aulas da Disciplina Sistemas e Redes:

Conteúdos teóricos e práticos disponibilizados na plataforma oficial da Faculdade de Ciência e Tecnologia da Universidade Fernando Pessoa, incluindo slides, exercícios, e vídeos das sessões ministradas pelo professor Vítor Sousa.

Estas fontes foram fundamentais para o sucesso do projeto, permitindo uma aplicação consistente das melhores práticas e o desenvolvimento eficiente da plataforma.