Estoril Track Manager

Relatório Complementar: Arquitetura de Ligação Backend-Frontend

Tomás Nogueira de Meneses Xavier 125438, Guilherme da Silva Gomes 125493
P1G7

22 de dezembro de 2025

Índice

1 Introdução

Este documento explica a forma como é feita a ligação entre a interface web (frontend) e a base de dados através do uso do servidor Flask na aplicação Estoril Track Manager. Tem como objetivo explicar a arquitetura da comunicação efetuada de maneira simples.

2 Tecnologias Utilizadas

Para a implementação da ligação foram utilizadas as seguintes tecnologias:

- **Frontend**: HTML, CSS e JavaScript foram usados para criar a interface do utilizador.
- **Template Engine**: Jinja2 foi uma biblioteca usada para a renderização dinâmica das páginas HTML no servidor.
- **Backend**: Flask, uma framework web em Python que processa pedidos e envia respostas.
- **Base de Dados**: Microsoft SQL Server para armazenamento de dados.
- **Driver de Conexão**: `pyodbc`, uma biblioteca Python que permite a criação de conexões diretas com a base de dados em SQL Server.

3 Estrutura do Projeto

O projeto está organizado em módulos que isolam responsabilidades e necessidades com o objetivo de melhorar a compartimentação do projeto:

- **app.py**: Ficheiro principal que inicializa a aplicação Flask e regista os blueprints de outros ficheiros.
- **utils/database.py**: Contém a função de conexão à base de dados, reutilizada por todos os módulos.
- **routes/auth.py**: Gere o login, registo, logout e definições de conta.
- **routes/public.py**: Ficheiro que possui rotas acessíveis a qualquer utilizador(pilotos,equipas,recordes e eventos).
- **routes/tecnico_pista.py, routes/diretor_equipa.py, routes/diretor_corrida.py**: Conjunto de ficheiros que identificam e implementam as funcionalidades e rotas de cada um dos roles.
- **templates/**: Contém todos os ficheiros HTML que são renderizados pelo servidor.
- **static/css/**: Contém as folhas de estilo CSS organizadas por componente.

4 Gestão de Conexões à Base de Dados

A conexão à base de dados é gerida através de um padrão chamado "context manager"que garante:

1. Abertura automática da conexão quando necessária.
2. Fecho automático da conexão após utilização, mesmo em caso de erro.
3. Rollback automático em caso de falha durante uma operação.

Todas as rotas utilizam este mecanismo para aceder à base de dados de forma segura e consistente. A conexão é estabelecida apenas durante o tempo necessário para executar a operação, sendo libertada imediatamente após.

5 Comunicação entre Frontend e Backend

Na comunicação entre Frontend e Backend existem dois sentidos para a comunicação:

5.1 Renderização no Servidor (Server-Side Rendering)

Este método é utilizado para carregar páginas completas:

- O browser faz um pedido GET a uma rota (exemplo: `/pilots`).
- O Flask recebe o pedido e executa a operação necessária na base de dados.
- Os dados obtidos são passados a um template HTML.
- O template Jinja2 processa os dados e gera HTML dinâmico.
- O HTML completo é enviado ao browser para apresentação.

5.2 API REST (Comunicação Assíncrona)

Já este método é utilizado de maneira a concluir operações dinâmicas sem haver a necessidade de recarregar a página:

- O JavaScript no browser faz um pedido à API (GET, POST, PUT ou DELETE).
- O pedido inclui dados em formato JSON quando necessário.
- O Flask processa o pedido e executa a operação na base de dados.
- O servidor responde com JSON indicando sucesso ou erro.
- O JavaScript atualiza a página conforme o resultado.

6 Sistema de Autenticação

A autenticação funciona através de sessões Flask:

- No momento em que o utilizador faz login, as credenciais são verificadas na base de dados.
- A password é comparada após aplicação de hash SHA256.
- Em caso de sucesso, são guardados na sessão: estado de login, ID do utilizador, username e role.
- Cada pedido subsequente verifica se a sessão contém um utilizador autenticado.
- O logout limpa a sessão e redireciona para a página inicial.

7 Fluxo de Dados

7.1 Da Base de Dados para o Ecrã

O Flask executa uma Stored Procedure ou query; os resultados são devolvidos como lista de tuplos. No template HTML, o Jinja2 trabalhas com os dados que são devolvidos e gera o HTML correspondente.

7.2 Do Formulário para a Base de Dados

O utilizador preenche um formulário; o JavaScript captura a submissão e envia via `fetch()` em JSON. O Flask extrai os valores, executa a Stored Procedure e devolve o resultado em JSON para o browser processar.

8 Conclusão

A arquitetura implementada garante uma interface simples e modular com uma separação clara entre interface e armazenamento.