ADVANCED BUSINESS DEVELOPMENT WITH .NET

Nome do aplicativo: Aiury

João Victor Madella – RM 561007

Nathália Mantovani de Falco — RM 99904

Renato de Angelo – RM 560585

Introdução

A Sprint 2 do projeto Aiury teve como foco o desenvolvimento da camada Web da aplicação, evoluindo a solução a partir da modelagem apresentada na etapa anterior. Nesta sprint, o objetivo principal foi disponibilizar uma interface funcional baseada no padrão MVC, permitindo interação completa com os dados do sistema por meio de operações de cadastro, consulta, edição e exclusão. Além da interface visual, foi implementada uma API com suporte a pesquisa, filtros, ordenação e navegação paginada, tornando possível o acesso aos dados tanto por usuários quanto por aplicações externas.

Essa etapa consolida a aplicação como um sistema executável e navegável, preparando a solução para fases futuras que incluirão persistência em banco de dados, autenticação, monitoramento e integração com outros ambientes tecnológicos envolvidos no projeto original. A sprint também atendeu aos requisitos formais da disciplina, com foco em boas práticas de arquitetura, separação de camadas e uso de recursos do .NET.

ASP.NET Core MVC / Views e Layouts

A camada Web do sistema foi desenvolvida utilizando o padrão MVC, com configuração de rotas que permite acesso estruturado às páginas principais da aplicação. Além da rota padrão, foram incluídas rotas personalizadas para simplificar a navegação, tornando possível acessar listas e detalhes de forma direta e organizada.

O sistema conta com um layout visual unificado, incluindo cabeçalho, rodapé e barra de navegação, aplicados em todas as páginas. A interface foi construída utilizando componentes visuais responsivos, com estilização baseada em Bootstrap. Esse layout fornece identidade visual consistente, com navegação clara entre as seções do sistema.

As páginas da aplicação contemplam todas as operações de cadastro, leitura, edição e exclusão referentes às entidades Ajudantes e Cidades, seguindo o fluxo completo de CRUD. Cada formulário possui validação automática em campo, garantindo que os dados enviados respeitem os critérios definidos no domínio da aplicação. A validação é realizada tanto no navegador quanto no envio da requisição, impedindo registro de valores inválidos.

Para assegurar a separação entre a camada de apresentação e a lógica da aplicação, foram utilizadas ViewModels. Essas estruturas atuam como intermediárias entre o formulário e o domínio, garantindo que apenas os dados necessários sejam manipulados, além de centralizar regras de validação e exibição. O uso de ViewModels evita exposição direta do modelo de domínio e reduz risco de inconsistências ou sobrecarga de dados na interface.

Minimal API / Web API

Além da camada visual, o sistema disponibiliza uma API com rotas destinadas à consulta dos dados das entidades Ajudantes e Cidades. Para cada domínio foi criada uma rota de pesquisa que permite filtragem, ordenação e paginação dos resultados, possibilitando uso eficiente em qualquer aplicação cliente, como uma interface web, aplicativo móvel ou integração externa.

As respostas da API incluem metadados relacionados à navegação entre páginas de resultados, como número total de registros, quantidade de páginas e posição atual. Isso permite que a aplicação consumidora controle a navegação sem necessidade de cálculos adicionais.

Foi implementado também o padrão HATEOAS, que consiste na inclusão de links navegáveis dentro da resposta da API. Esses links orientam o cliente a acessar a próxima página, retornar à anterior, ir para o início ou final da lista, ou ainda consultar a própria rota atual. Esse formato torna a API autodocumentada e facilita automatização no consumo.

Controllers ou Minimal API

A aplicação conta com controladores responsáveis por gerenciar todas as operações realizadas na interface web. Para cada uma das entidades da solução, foi implementado o conjunto completo de ações necessárias para cadastro, listagem, detalhamento, atualização e exclusão. Todas as ações relacionadas ao envio de dados utilizam validação para impedir gravação incorreta ou incompleta.

As camadas de controle foram estruturadas seguindo boas práticas, com separação clara entre lógica de apresentação, validação e armazenamento dos dados. A persistência é realizada por meio de um repositório que armazena temporariamente as informações em memória, conforme exigência desta etapa do projeto. Isso permite que o sistema funcione integralmente sem banco de dados, mantendo a estrutura pronta para substituição futura por persistência definitiva sem alterações na interface ou API.

A implementação dos controladores segue o fluxo recomendado pelo padrão MVC, incluindo confirmação de exclusão, redirecionamento adequado após operações concluídas e tratamento de erros como registros inexistentes ou parâmetros inválidos. A lógica de API foi mantida separada da interface, garantindo que a aplicação possa evoluir para uso híbrido ou multicanal sem necessidade de refatoração estrutural.

Conclusão

A Sprint 2 resultou na implementação completa da camada Web do projeto, permitindo interação com o sistema tanto por meio de interface visual quanto por API estruturada. Os requisitos funcionais e arquiteturais foram atendidos, com a criação de CRUDs completos, estrutura de navegação, validações, ViewModels, rotas personalizadas e serviços de consulta remota com HATEOAS.

A solução entregue encontra-se preparada para expansão nas próximas etapas, nas quais serão adicionados persistência em banco de dados, autenticação de usuários, comunicação com outros sistemas e integração com aplicações móveis. Assim, esta sprint estabelece a base operável da aplicação, transformando a especificação inicial em um sistema funcional e demonstrável.