Documento de Projeto de Sistema

English For All Time

Vitória, ES

2025

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Matheus De Oliveira	27/05/2025	Versão inicial.

1 Introdução

Este documento apresenta o projeto (*design*) do sistema *English For All Time*.

É um plataforma de ensino de inglês onde o professor-administrador tem controle total: ele pode cadastrar seus alunos e adicionar/criar cursos diretamente no sistema, organizando-os em módulos com vídeos (via links do YouTube não listados), materiais em PDF e exercícios. A plataforma oferece um painel intuitivo para o dono gerenciar tanto os usuários quanto os conteúdos publicados, permitindo atualizações rápidas e personalizadas, sem depender de terceiros. E para os alunos eles terão uma página com todos os conteúdos publicados pelo professor.

Além desta introdução, este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação do sistema; a Seção 3 apresenta a arquitetura de software; por fim, a Seção 4 apresenta os modelos FrameWeb que descrevem os componentes da arquitetura.

2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas.

Tecnologia	**Versão**	**Descrição**	**Propósito**
React.js	18+	Biblioteca JavaScript para interfaces dinâmicas	Frontend responsivo para alunos e admin
TypeScript	5.x	Superset tipado de JavaScript	Frontend responsivo para alunos e admin
Spring Boot	3.2.x (Java 17)	Framework backend Java	API RESTful segura e escalável
Spring Web MVC	6.1.x	Módulo para construção de APIs REST	Rotas HTTP e serialização JSON
Spring Security	6.1.x	Autenticação e autorização	Controle de acesso (JWT)
Spring Data JPA	3.1.x	Persistência com Hibernate	Operações de banco de dados (PostgreSQL)
PostgreSQL	15+	Banco de dados relacional	Armazenar usuários, cursos e progresso
jjwt	0.12.x	Biblioteca para JWT	Geração/validação de tokens
Lombok	1.18.x	Redução de boilerplate em classes Java	Getters/Setters automáticos
Hibernate Validator	8.0.x	Validação de dados em DTOs	Validar entradas de API (ex.: @Email, @NotBlank)

Tecnologia	Versão	Descrição	Propósito
React Router	6.x	Roteamento no frontend	Navegação entre páginas
Axios	1.x	Cliente HTTP para frontend	Consumir API do backend
Material UI	5.x	Biblioteca de componentes UI	Design consistente e responsivo

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	Versão	Descrição	Propósito
PgAdmin 4	7.x+	Interface gráfica para PostgreSQL	Gerenciar visualmente o banco de dados
IntelliJ IDEA	2023.2+	IDE para Java/Spring Boot	Desenvolvimento backend com debug integrado
VS Code	1.80+	Editor para React/TypeScript	Codificação do frontend com extensões úteis
Postman	10+	Teste de APIs	Validar endpoints do Spring Boot
Git	2.40+	Controle de versão	Gerenciar colaboração no código
Visual Paradigm	17.2	Criação de diagramas UML	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
FrameWeb	-	Plugin do Visual Paradigm	Auxilia na criação da estrutura FrameWeb para o desenvolvimento dos modelos.
TeX Live	2018	Implementação do LaTeX	Documentação do projeto arquitetural do sistema.
TeXstudio	4.8.7	Editor de LaTeX.	Escrita da documentação do sistema.
Apache Maven	3.5	Ferramenta de gerência/construção de projetos de software.	Obtenção e integração das dependências do projeto.

3 Arquitetura de Software

A Figura 1 mostra a arquitetura do sistema *English For All Time*. Ela é baseada em uma combinação dos estilos arquitetônicos Camadas e Partições.

A arquitetura do sistema é estruturada em três camadas principais, cada uma com responsabilidades distintas e bem definidas, conforme ilustrado na Figura 1.

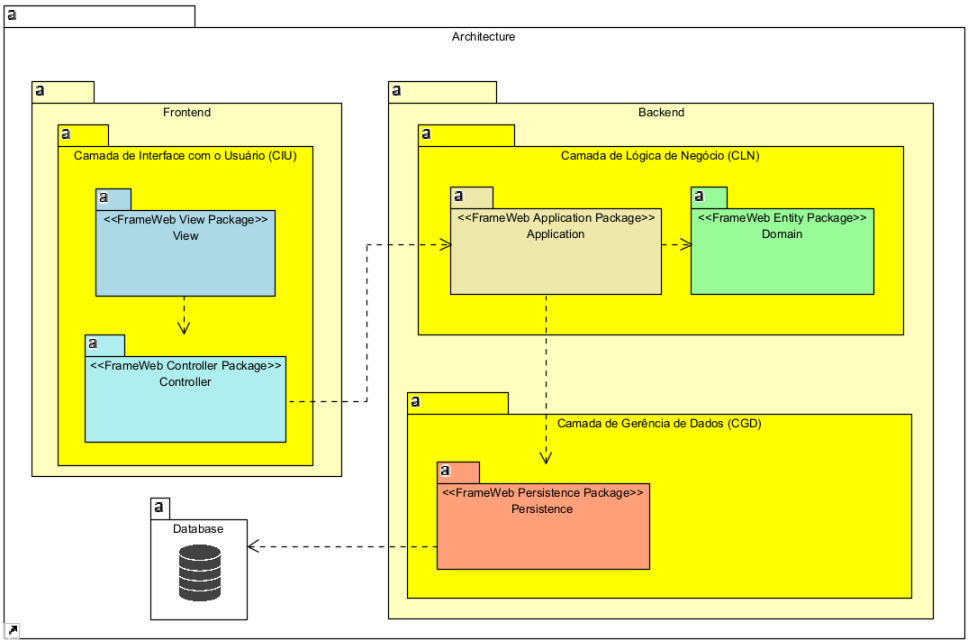

Figura 1 – Arquitetura de Software.

3.1 Camada de Interface com o Usuário (CIU)

Localizada no **Frontend**, essa camada é responsável pela interação com os usuários, exibindo dados e capturando ações externas. Ela adota o padrão **MVCS (Model-View-Controller-Service)** e está dividida em dois pacotes principais:

- **View (`«FrameWeb View Package»`)**: Contém os elementos da camada de apresentação, como páginas web, folhas de estilo, imagens e scripts do lado cliente.
- **Controller (`«FrameWeb Controller Package»`)**: Coordena a interação entre a visão e os serviços da aplicação localizados na camada de lógica de negócio (CLN). A dependência entre os componentes é unidirecional: a *view* depende do *controller*, e este depende da lógica de aplicação.

3.2 Camada de Lógica de Negócio (CLN)

Localizada no **Backend**, essa camada implementa as regras de negócio do sistema. Ela segue o padrão **Camada de Serviço**, conforme definido por Fowler (2002), e é composta por dois pacotes:

- **Application (`«FrameWeb Application Package»`)**: Contém as classes responsáveis pela orquestração das funcionalidades do sistema, utilizando objetos de domínio e interagindo com a camada de persistência.

- **Domain (`«FrameWeb Entity Package»`)**: Reúne as classes que representam os elementos centrais do domínio do problema. Tais entidades são manipuladas pela *application* para a implementação das regras de negócio.

3.3 Camada de Gerência de Dados (CGD)

Também localizada no **Backend**, essa camada é responsável pela persistência dos dados e segue o padrão **Data Access Object (DAO)**. Contém o seguinte pacote:

- **Persistence (`«FrameWeb Persistence Package»`)**: Inclui as classes responsáveis pelas operações de acesso a dados, realizando mapeamento objeto-relacional (ORM) para persistir as entidades do domínio no banco de dados relacional.

3.4 Integração entre as Camadas

- A **View** se comunica com o **Controller**.
- O **Controller** interage com a camada **Application**.
- A **Application** manipula entidades do **Domain** e utiliza a camada **Persistence** para realizar a persistência dos dados.
- A **Persistence** acessa diretamente o **banco de dados relacional**.

4 Modelagem FrameWeb

English For All Time é um sistema Web cuja arquitetura utiliza *frameworks* comuns no desenvolvimento para esta plataforma. Desta forma, o sistema pode ser modelado utilizando a abordagem FrameWeb (SOUZA, 2020).

A Tabela 3 indica os *frameworks* presentes na arquitetura do sistema que se encaixam em cada uma das categorias de *frameworks* que FrameWeb dá suporte. Em seguida, os modelos FrameWeb são apresentados para cada camada da arquitetura.

Tabela 3 – *Frameworks* da arquitetura do sistema separados por categoria.

Categoria de *Framework*	***Framework* Utilizado**
Controlador Frontal	Spring MVC
Injeção de Dependências	Spring Framework

Categoria de *Framework*	***Framework* Utilizado**
Mapeamento Objeto/Relacional	Spring Data JPA
Segurança	Spring Security

4.1 Camada de Negócio

A figura 2 mostra o projeto da Camada de Negócio (pacote domain) do sistema.

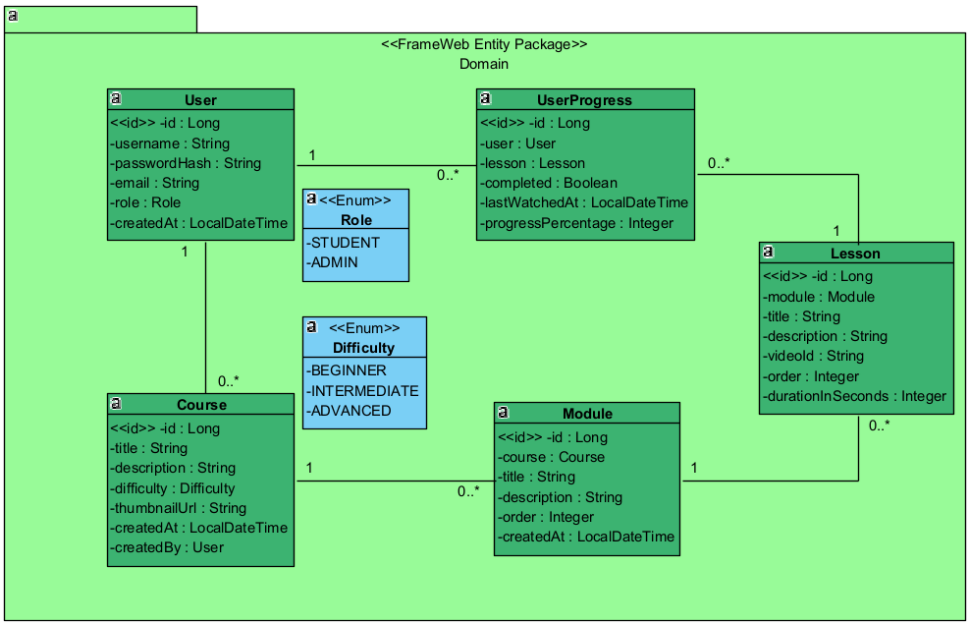

Figura 2 – Projeto da Camada de Negócio (domain)

Dentro do pacote domain, existem as entidades em verde escuro que são usadas como base das regras de negócio e da persistência de dados; enquanto as classes em azul claro, são tipos enumerados que representam classificações de usuário dentro do sistema (Role) e identificação da dificuldade de um curso (Difficulty).

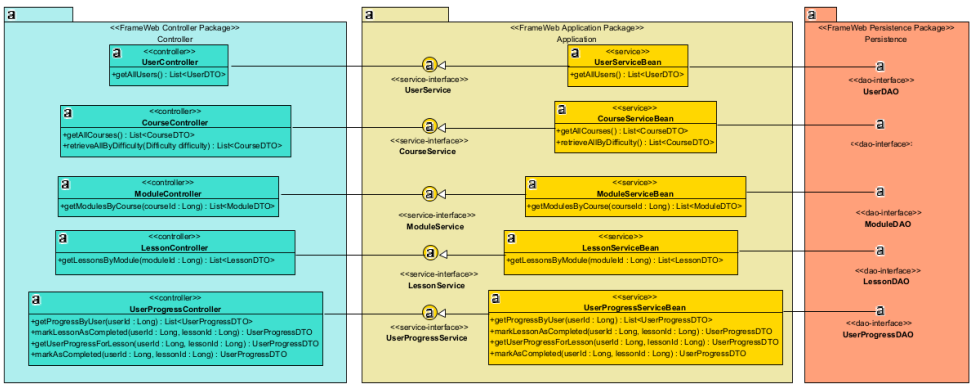

Figura 3 – Projeto da Camada de Negócio (application)

A arquitetura apresentada na Figura 3 ilustra a integração entre os pacotes Controller, Application e Persistence, os quais, em conjunto, viabilizam o retorno ao frontend dos resultados decorrentes da aplicação das regras de negócio solicitadas.

Vale destacar que, por se tratarem de operações básicas comuns a todas as entidades, as classes responsáveis pelas operações CRUD (create, read, update e delete) não foram explicitamente representadas na figura. Além disso, embora os métodos dos controllers retornem os dados encapsulados em objetos do tipo ResponseEntity, esse detalhe também foi omitido na representação para manter o diagrama mais conciso e focado na estrutura geral do fluxo de dados e responsabilidades.

4.2 Camada de Acesso a Dados

A figura 4 mostra o projeto da Camada de Acesso a Dados (pacote persistence) do sistema.

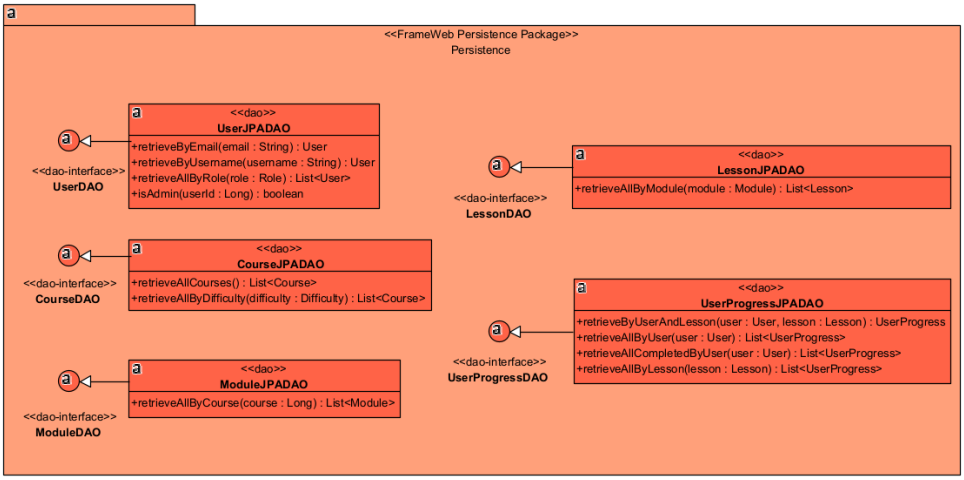

Figura 4 – Projeto de Acesso a Dados (persistence)

Nessa camada, como estará sendo utilizado o Spring Data JPA, por padrão ele já tem as buscas como create, read, update e delete (CRUD) das entidades básicas. Então, dentro de cada JPADAO do pacote persistence terão seus métodos personalizados que variam de acordo com a necessidade de acesso a dados do backend para aplicar suas regras de negócio que vão ser passadas ao frontend.

Enquanto o Spring Data JPA, há os métodos personalizados que fazem:

- **UserJPADAO**: autenticação, recuperação de senha, validação de cadastro, cadastro/validação de nomes únicos, listar todos os alunos ou administradores e garantir que o usuário está acessando a área administrativa.
- **CourseJPADAO**: visualizar detalhes de um curso, vincular módulos a um curso, filtrar cursos por dificuldade.
- **ModuleJPADAO**: listar todos os módulos de um curso quando o aluno entra no curso.

- **LessonJPADAO**: listar as aulas dentro de um módulo, buscar uma aula específica em uma ordem (ex: próxima aula).
- **UserProgressJPADAO**: verificar se o aluno já completou a lição, listar progresso completo do aluno (quais aulas ele já viu), exibir o número de lições concluídas e estatísticas da aula: quantos alunos viram aquela lição.

4.3 Camada de Apresentação

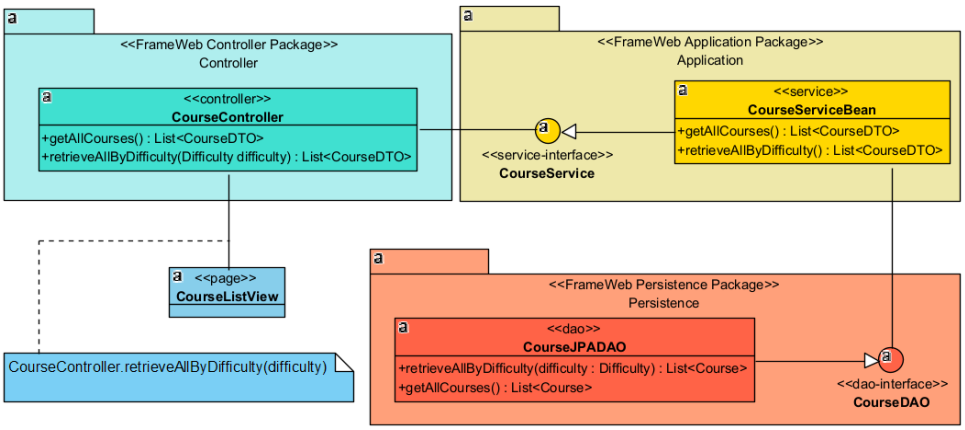

Figura 5 – Modelo de Navegação (Visualizar Cursos)

A imagem 5 representa a arquitetura da funcionalidade de consulta de cursos disponíveis em uma aplicação web estruturada segundo o padrão FrameWeb.

A operação é iniciada na camada de controle, com o método retrieveAllByDifficulty(difficulty) do CourseController, responsável por receber a requisição da interface de usuário, mais especificamente da página CourseListView, e encaminhá-la para a camada de aplicação. Nessa camada, o CourseServiceBean, implementando a interface CourseService, executa a lógica de negócio associada à recuperação de cursos conforme o nível de dificuldade selecionado, garantindo a aplicação das regras do domínio.

O service então delega a consulta à camada de persistência, onde o CourseJPADAO, implementação da interface CourseDAO, executa o método retrieveAllByDifficulty, realizando a busca no banco de dados e retornando uma lista de entidades Course. Além disso, a arquitetura contempla também o método getAllCourses, que possibilita a listagem completa dos cursos sem filtragem, percorrendo a mesma estrutura de camadas.

A imagem 6 representa a arquitetura da funcionalidade de consulta dos módulos de um curso em uma aplicação web estruturada segundo o padrão FrameWeb.

A operação é iniciada na camada de controle, com o método getModulesByCourse(courseId) do ModuleController, responsável por receber a requisição da interface de usuário, mais especificamente da página ModuleListView, e encaminhá-la para a camada de

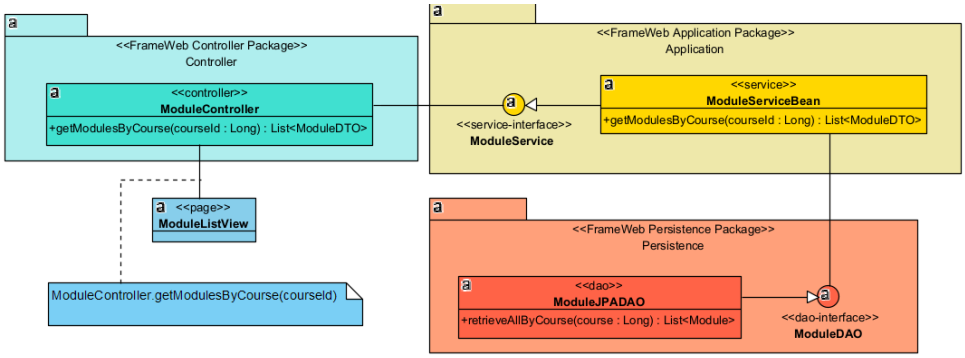

Figura 6 – Modelo de Navegação (Visualizar Módulos de um Curso)

aplicação. Nessa camada, o ModuleServiceBean, implementando a interface ModuleService, executa a lógica de negócio associada à recuperação dos módulos pertencentes ao curso identificado pelo parâmetro courseId, garantindo a aplicação das regras do domínio.

O service então delega a consulta à camada de persistência, onde o ModuleJPADAO, implementação da interface ModuleDAO, executa o método retrieveAllByCourse, realizando a busca no banco de dados e retornando uma lista de entidades Module.

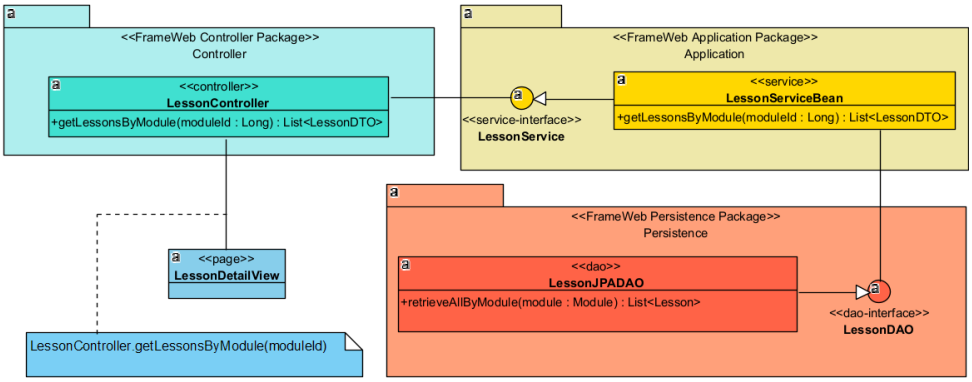

Figura 7 – Modelo de Navegação (Marcar Lição como Concluída)

A imagem 7 representa a arquitetura da funcionalidade de consulta das aulas pertencentes a um módulo em uma aplicação web estruturada segundo o padrão FrameWeb.

A operação tem início na camada de controle, com o método getLessonsByModule(moduleId) do LessonController, que recebe a requisição proveniente da interface de usuário, mais especificamente da página LessonDetailView, e a encaminha para a camada de aplicação. Nessa camada, o LessonServiceBean, que implementa a interface LessonService, executa a lógica de negócio necessária para recuperar as aulas vinculadas ao módulo identificado pelo parâmetro moduleId, assegurando o correto funcionamento conforme as regras do domínio.

Em seguida, o service delega a operação à camada de persistência, onde o LessonJPADAO, implementação da interface LessonDAO, executa o método retrieveAllByModule, realizando a consulta no banco de dados e retornando uma lista de entidades Lesson.

Referências

FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Nenhuma citação no texto.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <http://purl.org/nemo/celebratingfalbo>. Citado na página 5.