O método FrameWeb aplicado ao sistema Vital

Gabriel Santa Clara Ucelli e Luiz Felipe Ferreira Mai

27 de Julho de 2017

1 O método FrameWeb

FrameWeb é um método de projeto para construção de sistemas de informação Web (Web Information Systems – WISs) baseado em *frameworks*. O método assume que determinados tipos de frameworks serão utilizados durante a construção da aplicação, define uma arquitetura básica para o WIS e propõe modelos de projeto que se aproximam da implementação do sistema usando esses *frameworks*. Para uma melhor utilização de FrameWeb, espera-se que sejam construídos diagramas de casos de uso e de classes de domínio (ou similares) durante as fases de Requisitos e Análise.

O FrameWeb define uma arquitetura lógica padrão para WISs baseada no padrão arquitetônico *Service Layer* (Camada de Serviço). O sistema é dividido em três grandes camadas:

- **Camada de Negócio *(Business Tier)*:** responsável pelas funcionalidades relacionadas às regras de negócio da aplicação. Esta camada é particionada em duas: Lógica de Domínio (Domain) e Lógica de Aplicação (Application);
- **Camada de Apresentação *(Presentation Tier)*:** responsável por funcionalidades de interface com o usuário (incluindo as telas que serão apresentadas a ele). Esta camada, segundo o padrão proposto, é também particionada em duas outras: Visão (View) e Controle (Control);
- **Camada de Acesso a Dados *(Data Access Tier)*:** responsável por funcionalidades relacionadas à persistência de dados.

Para o método, a fase de Projeto concentra as propostas principais do método: (i) definição de uma arquitetura padrão que divide o sistema em camadas, de modo a se integrar bem com os *frameworks* utilizados; (ii) proposta de um conjunto de modelos de projeto que trazem conceitos utilizados

pelos *frameworks* para esta fase do processo por meio da definição de uma linguagem específica de domínio que faz com que os diagramas fiquem mais próximos da implementação.

Para representar componentes típicos da plataforma Web e dos *frameworks* utilizados, o FrameWeb estende o meta-modelo da UML, especificando, assim, uma sintaxe própria. Com isso, é possível utilizá-lo para a construção de diagramas de quatro tipos:

- **Modelo de Entidades:** é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional;
- **Modelo de Persistência:** é um diagrama de classes da UML que representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio.
- **Modelo de Navegação:** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e *controllers*.
- **Modelo de Aplicação:** é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências.

2 O sistema Vital

O Vital foi desenvolvido com o intuito de estabelecer uma ligação direta entre pacientes e médicos de diversos hospitais. Nele, espera-se que pacientes possam marcar consulta com um médico no horário desejado e que, após realizada a consulta, este médico possa registrar um diagnóstico referente àquela consulta, incluindo patologia, medicamentos etc.

A fim de demonstrar a arquitetura utilizada no desenvolvimento do sistema, serão utilizados os modelos propostos pelo método FrameWeb: a seção 2.1 representa o modelo de Entidades do sistema, enquanto a seção 2.2 apresenta o a seção 2.1 aborda o modelo de Aplicação e, por fim, a seção 2.1 mostra o modelo de Navegação referente às operações básicas com pacientes.

2.1 Modelo de Entidades

Nesta seção é apresentado o Modelos de Entidades desenvolvido para o sistema Vital.

Diferentemente da abordagem FrameWeb original proposta em 2007, todos os atributos que não podem ser nulos tiveram a tag `not null` omitida e os que pdoem tiveram a tag `null` acrescida de forma a diminuir a poluição visual com repetições desnecessárias.

Todas as classes de domínio estendem `PersistentObjectSupport` do pacote `JButler`, sendo que essa herança não é mostrada nos diagramas com o intuito de não poluí-los com várias associações.

A Figura 1 mostra o Modelo de Entidades para o sistema.

2.2 Modelo de Persistência

Nesta seção é apresentado o Modelos de Entidades desenvolvido para o sistema Vital.

Como pode-se perceber, o sistema adota o padrão DAO para persistência: uma sugestão do FrameWeb para simplificar o processo de desenvolvimento de software.

Todas as interfaces do modelo herdam de `BaseDAO` e, todas as classe, de `BaseJPADAO`. Essas heranças não são representadas explicitamente no modelo a fim de evitar poluição visual.

A Figura 2 mostra o Modelo de Persistência para o sistema.

2.3 Modelo de Aplicação

Nesta seção é apresentado o Modelos de Aplicação desenvolvido para o sistema Vital.

Todas as classes de aplicação que são de casos de uso cadastrais estendem de CrudServiceBean do pacote `jbutler`. Da mesma forma que no diagrama anterior, essa herança não é representada com o intuito de não polui o diagrama com várias associações.

Os casos de uso para *Fazer Login* e *Criar Conta* foram adicionados dentro das classes `LoginService` e `RegistrationService`.

A Figura 3 mostra o Modelo de Aplicação para o sistema.

2.4 Modelo de Apresentação

Nesta seção é apresentado o Modelos de Apresentação desenvolvido para o sistema Vital.

Embora outras classes realizem as operações de *CRUD*, decidimos utilizar a classe `Patient` como exemplo. A Figura 4 mostra o Modelo de Apresentação para os pacientes do sistema, incluindo listagem, criação, edição, detalhamento e deleção.

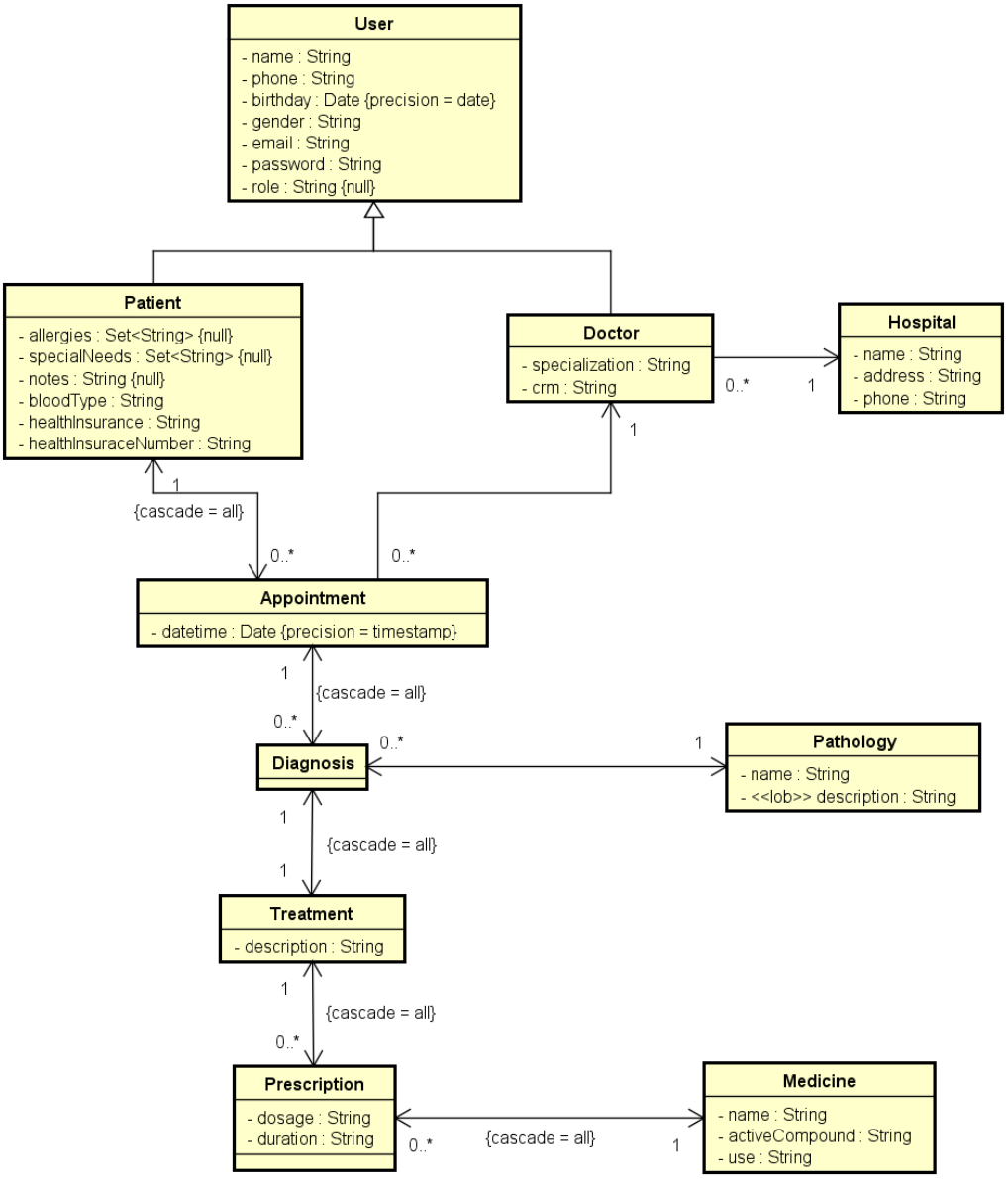

Figura 1: Modelo de Entidades para o sistema Vital

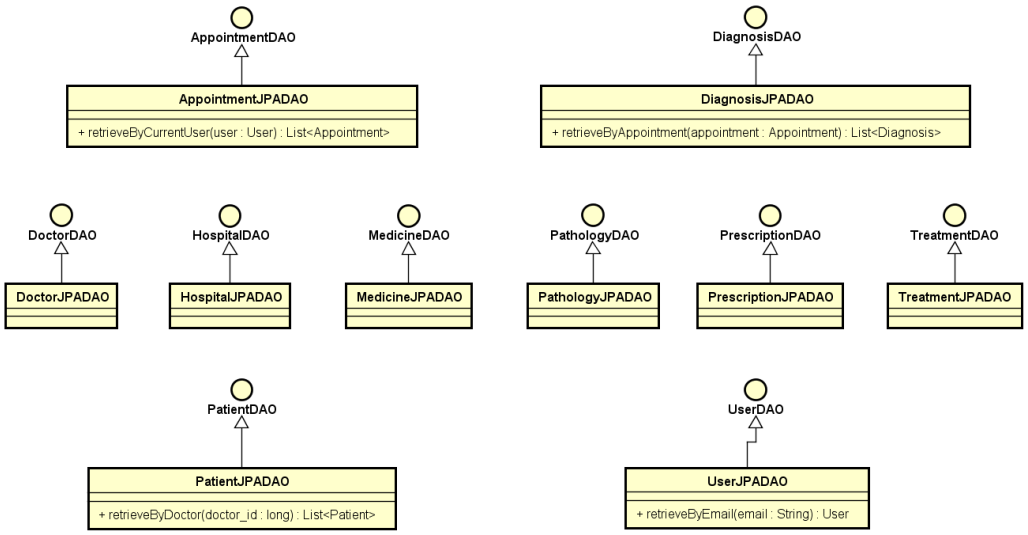

Figura 2: Modelo de Persistência para o sistema Vital

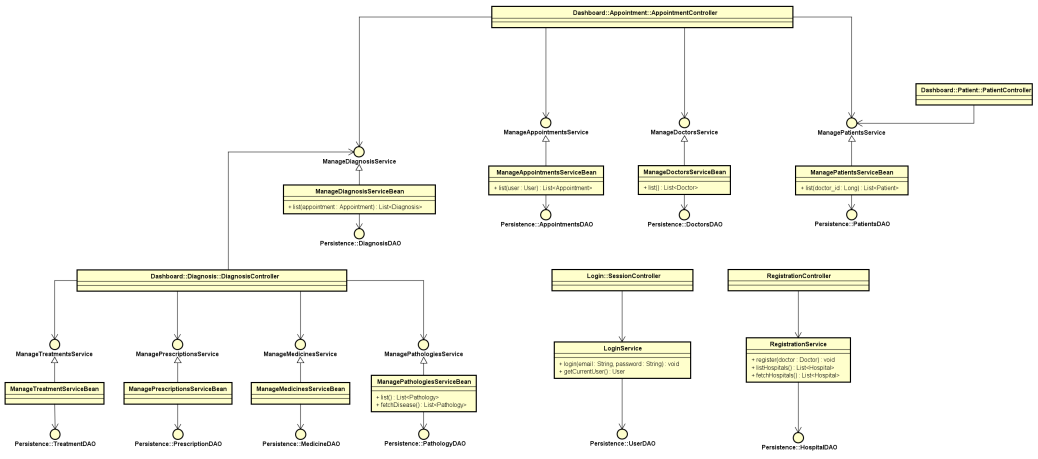

Figura 3: Modelo de Aplicação para o sistema Vital

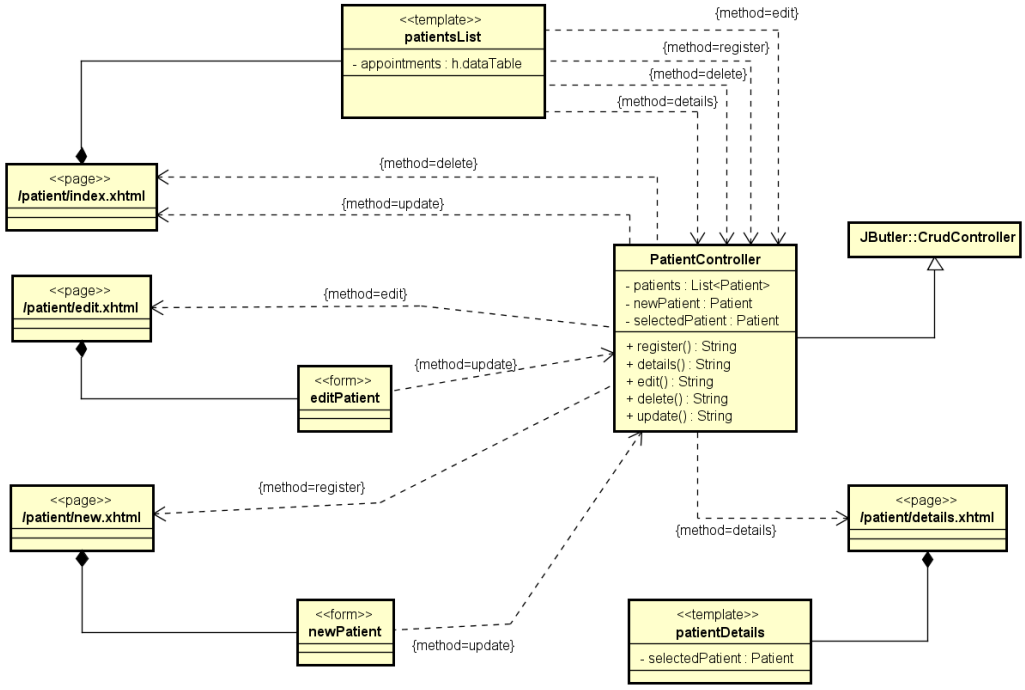

Figura 4: Modelo de Apresentação para o sistema Vital