

Segurança Informática em

Redes e Sistemas

Medical Records Database

(Med-DB)

Afonso Guilherme Falardo Romeira Garcia - 70001

José Luis Domingues Góis – 79261

João Paulo Marques dos Santos – 76276

**MEIC-ALAMEDA**

**2013/2014**

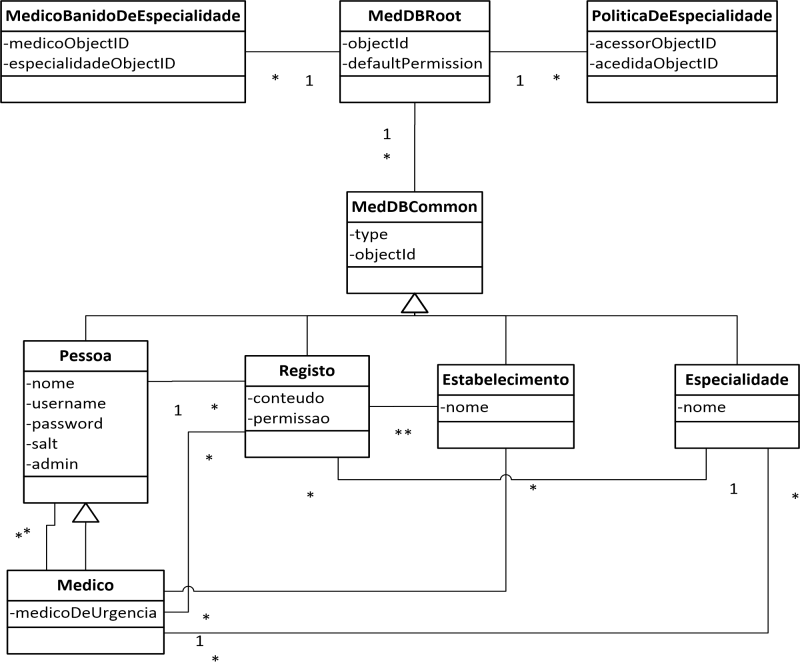
# Introdução

O projecto tem como objectivo a criação de um sistema electrónico de armazenamento de registos médicos com base nos seguintes princípios:

* Criação/estruturação de uma base de dados médica;
* Definição de políticas de acesso com base em autenticação;
* Garantia de Confidencialidade da informação;
* Definição de políticas de acesso aos registos com base em roles personalizáveis e dinâmicos;

# Modelo de Dominio

Tendo em conta as características e especificidades do nosso sistema, definimos a seguinte estrutura como modelo de domínio:

****

# Implementações de Segurança

No âmbito das implementações de segurança, a aplicação oferece garantias de autenticação e confidencialidade, cujos mecanismos aplicados serão descritos nas secções seguintes.

## Autenticação

Em termos de autenticação todos os utilizadores do sistema necessitam de um nome de utilizador(obrigatoriamente único) e uma palavra-chave. Para definição da palavra-chave existem determinadas regras que têm de ser cumpridas:

* A palavra-chave deve conter entre 6 e 18 caracteres, pelo menos 1 dígito e pelo menos um caracter não alfanumérico;
* A palavra-chave não deve conter espaços em branco, sequências de dígitos ou alfanuméricas, três caracteres seguidos repetidos e sequências do tipo “QWERTY”.

Para o último tópico referido é ainda feito uma análise com base em três dicionários contendo as palavras e nomes mais comuns e lugares/localidades em Inglês (por ser de fácil obtenção na *Internet*).

A palavra-chaveé armazenada na base de dados recorrendo à função de *hash* SHA1 e a um valor aleatório (*salt*) e é armazenado o resultado da seguinte função: *SHA1(palavra-chave¦¦salt)*. O *salt* é calculado de forma aleatória recorrendo ao *SecureRandom* do *Java*.

## Confidencialidade

Para o caso da confidencialidade, os dados críticos da aplicação são cifrados com a chave simétrica do sistema, sendo esta gerada automaticamente pela aplicação aquando da inicialização da base de dados da mesma. Consideram-se dados críticos os conteúdos dos registos.

Caso um utilizador pretenda aceder a um registo, o sistema verifica se o mesmo tem permissões para tal e em caso afirmativo é devolvido o registo decifrado.

Para verificar se o utilizador tem permissão para aceder ao recurso a aplicação implementa permissões baseadas em *roles* e permissões estáticas. Os *roles* são definidos pela especialidade do médico.

## PolÍticas

O sistema implementa um conjunto de 3 políticas, que definem o acesso aos registos por parte dos utilizadores. Essas políticas permitem especificar qual o conjunto de permissões com o qual o registo é criado, que especialidades podem aceder aos registos de uma dada especialidade e se um médico não pode aceder a determinada especialidade.

Qualquer uma das 3 políticas é alterável em *runtime* através da interface de administração do *Med-DB*, ao qual têm acesso os utilizadores com poderes de administração.

## Permissões

O sistema implementa as seguintes permissões:

* *PermissaoMedicoDaEspecialidade*: permite o acesso ao registo por parte dos médicos da especialidade do registo;
* *PermissaoMedicoDaPessoa*: permite o acesso ao registo por parte dos médicos do paciente do registo;
* *PermissaoMedicoDeUrgencia*: permite o acesso ao registo por parte dos médicos que trabalham em serviços de urgência;
* *PermissaoMedicoDoEstabelecimento*: permite o acesso ao registo por parte dos médicos do estabelecimento do registo;
* *PermissaoMedicoDoRegisto*: permite o acesso ao registo por parte do médico do registo;
* *PermissaoPacienteDoRegisto*: permite o acesso ao registo por parte do paciente do registo;
* *PermissaoPublica*: permite o acesso ao registo por parte de qualquer médico ou paciente.

As seguintes permissões implementam as políticas referidas:

* *PermissaoPoliticaDeEspecialidade*: permite o acesso ao registo por parte de médicos de uma especialidade com acesso aos registos da especialidade do registo;
* *PermissaoMedico*: permite o acesso ao registo por parte de um médico.

As permissões acima referidas podem ser compostas com os operadores booleanos *e* e *ou* (*PermissaoELogico* e *PermissaoOuLogico*, respectivamente) ou negadas (*PermissaoNaoLogico*).

As permissões são definidas de acordo com a gramática em anexo, que permite fazer o *parse* de uma *string* que define as permissões.

# Resistência a Ataques

O sistema, apesar de apresentar vulnerabilidades (ataques de força bruta, por exemplo), é seguro em alguns aspectos, tais como os abaixo apresentados.

## SQL Injection

A aplicação utiliza a *Fenix Framework* como forma de abstracção do modelo de domínio e esta *framework* não é vulnerável a este tipo de ataques.

## Cross-Site Scripting

Uma vez que a aplicação foi desenvolvida para correr em ambiente local e não distribuído via web, este ataque não e realizável.

## Stack e Buffer Overflow

A linguagem Java recorre a alocação dinâmica de memória e referencia tudo através de ponteiros, não sendo por isso vulnerável ao ataque referido.

# Falhas

O sistema não permite a não-repudiação e autenticação de registos, uma qualidade de segurança desejável num sistema deste tipo pois não conseguimos implementar um mecanismo que o permitisse a tempo da entrega.

De forma semelhante, o sistema permite que seja efectuado um ataque às palavras-chave por força bruta e não possui mecanismos que permitam alterar a chave simétrica em caso de descoberta desta.

# Anexos

## como correr o projecto

Para correr o projecto é necessário seguir os passos descritos nesta secção.

### Software necessário

* Java 6 ou superior;
* MySQL 5 ou superior;
* Maven 3.0.5 ou superior;

### Compilação do Projecto

O projecto é compilado recorrendo ao seguinte comando:

mvn package

### Preparação da base de dados

É necessário criar um utilizador de acordo com os dados abaixo indicados, que tenha permissões de *DBManager* e tenha todas as permissões sobre *schemas* com o nome *med-db*.

|  |  |
| --- | --- |
| Utilizador | med-db |
| Palavra-chave | med-db-pass |

### Popular a base de dados

São disponibilizados um conjunto de dados de teste para execução do projecto. Para os colocar na base de dados basta executar o seguinte comando:

mvn exec:java –Dexec.mainClass=”pt.ist.sirs.application.PopulateDB”

O sistema irá questionar onde colocar a chave simétrica do *Med-DB*. Guarde esta localização para depois. Não coloque nenhum nome para o ficheiro, apenas a pasta onde o guardar (caminho completo desde / e terminar sempre com /).

### Correr a aplicação

A aplicação é corrida com o seguinte comando:

mvn exec:java –Dexec.mainClass=”pt.ist.sirs.application.MedDBApp” –Dexec.args=”<pasta da chave>”

Coloque em <pasta da chave> o caminho completo para a chave que guardou anteriormente.

O projecto está pronto a ser utilizado.

## tecnologias utilizadas

* Java 1.6
* MySQL 5
* Fenix Framework 1
* Maven 3
* AES
* ANTLR 4
* VT-Middleware 3
* Apache Commons Codec 1.7

## Gramática de PErmissões

/\* PARSER RULES \*/

b: OR BP s CP | AND BP s CP | NOT BP b\_1 CP | MDESP | MDP | MDU | MDEST | MDR | PDR | PPDE | PP;

b\_1: b;

s\_1: s;

s: b VR s\_1 | b;

/\* LEXER RULES \*/

OR: 'or';

AND: 'and';

NOT: 'not';

MDESP: 'mdesp';

MDP: 'mdp';

MDU: 'mdu';

MDEST: 'mdest';

MDR: 'mdr';

PDR: 'pdr';

PPDE: 'ppde';

PP: 'pp';

BP: '(';

CP: ')';

VR: ',';

INT: (DIGIT)+;

DIGIT: '0'..'9';