RELATÓRIO

Alunos: Andrew Henrique, Bruno Bastos, Victor Bueno

Tema: Sistema de Atendimento Médico Online

Disciplina: Desenvolvimento para Web 2

Turma: 647

Data: 22/06/2017

OBJETIVO:

Objetivou-se desenvolver um sistema que oferecesse serviços de cadastro de consultas, pacientes e médicos voluntários. Permitindo que os atendimentos fossem agendados e organizados virtualmente. Seria possível visualizar as informações dos indivíduos cadastrados, levando em consideração a relevância destas de acordo com o perfil(médico ou paciente), bem como os dados das consultas agendadas.

O uso do sistema não oferece custos aos usuários, pois considerou-se como parte do objetivo incial o desenvolvimento de uma aplicação de fácil acesso para qualquer tipo de usuário, incluindo pessoas que não possuem condição financeira, facilitando o acesso delas a atendimentos médicos, que muitas vezes têm um custo elevado.

BANCO DE DADOS

Hsqldb 2.3.4

Com uso de hibernate

BIBLIOTECAS UTILIZADAS:

Nesse projeto, foram utilizadas todas as bibliotecas do *hibernate 4.3.11(Final),* usadas para auxiliar a implementação do banco de dados em JPA. A biblioteca para conexão do hsqldb. E a biblioteca do JSF.

Lista detalhada:

Antlr-2.7.7.jar; classmate-1.3.0.jar; dom4j-1.6.1; hibernate-commons-annotations-4.0.5.Final.jar; hibernate-commons-annotations-5.0.1.Final.jar; hibernate-core-4.3.11.Final.jar; hibernate-core-5.2.10.Final.jar; hibernate-entitymanager-4.3.11.Final.jar; hibernate-envers-5.2.10.Final.jar; hibernate-jpa-2.1-api-1.0.0.Final.jar; hsqldb.jar; jandex-1.1.0.Final.jar; jandex-2.0.3.Final.jar; javassist-3.18.1-GA.jar; javassist-3.20.0-GA.jar; javax.faces-2.2.8-17.jar; jboss-logging-3.1.3.GA.jar; jboss-logging-3.3.0.Final.jar; jboss-logging-annotations-1.2.0.Beta1.jar, jboss-transaction-api\_1.2\_spec-1.0.0.Final.jar; jboss-transaction-api\_1.2\_spec-1.0.1.Final.jar;

FUNCIONAMENTO/USABILIDADE:

Para iniciar as atividades na ferramenta, primeiramente o usuário deverá se identificar, fornecendo o nome de usuário e a senha. O software usará as informações para consultar os registros no banco de dados, para então liberar as páginas autorizadas à pessoa em questão, sendo essas *home pages.* IssoOcorre na classe "AutenticadorBean", do pacote "socialdoc.controle", que, além de autenticar os dados do usuário, retorna a página principal de acordo com o tipo de usuário, verificado a partir do método "getTipoUsuario", na classe "UsuarioRepository"(pacote "socialdoc.repository").

Responsável por métodos de consultas e inserções no banco de dados das tabelas sobre usuários, a classe "UsuarioRepository" guarda métodos de adicionar e remover usuários, além de métodos para pesquisar determinados usuários.

Caso não tenha uma conta no sistema, o usuário poderá acessar as funções para cadastro de dados, tanto para pacientes quanto médicos. Ainda no pacote "socialdoc.controle", as classes "CadastrarMedicoBean" e "CadastrarPacienteBean" têm os métodos para cadastrar e atualizar os dados. O usuário apenas verá formulários com textos indicando as informações com as quais os campos deverão ser preenchidos, mostradas a ele através das *views.* "login", "registrar\_medico", "registrar\_paciente", em arquivos *.xhtml*.

As classes do pacote "socialdoc.controle", acessam os métodos e atributos das classes do pacote "socialdoc.model", que são Medico, Paciente Usuario, Administrador e PeriodoDisponivel. Paciente e Medico são classes que contem os atributos e métodos para acesso dos mesmos referente aos perfis de usuário médico e paciente. Elas herdam alguns atributos em comum de todos os usuários, contidos na classe "Usuario".

Uma vez tendo acessado sua página inicial, caso o indivíduo seja um médico, poderá ter acesso aos pacientes que buscam pelo seu atendimento, sendo também possível verificar as consultas marcadas atribuídas a esse médico. Ele também poderá alterar as informações cadastradas ao seu respeito, além de marcar consultas e atribuir a ele pacientes. Na pasta "views", as páginas que devem ser mostradas ao médico para que ele possa realizar tais ações constam na pasta "médico". "medico\_agendar", "medico\_consultas", "medico\_editar" e "medico\_marcar" são alguns exemplos de arquivos *xhtml* que levam o usuário a navegar pelas funções de médico, contidas nas classes supracitadas.

Os médicos podem cadastrar datas e horas que ele está disponível, aí o usuário pode escolher uma dessas datas para criar as consultas.

Já o "usuário paciente", assim como o médico, terá a possibilidade de cadastrar seus dados e editá-los, além de marcar consultas e colocar seu nome entre os pacientes a serem atendimentos por determinado médico, através das *views* contidas na pasta "paciente", dentro de "views". Esse diretório hospeda páginas semelhantes às páginas disponíveis para médicos, porém são versões para os pacientes, dando-lhes muitas das opções existentes para os profissionais da área de saúde.

Conforme foi mencionado em partes anteriores, a aplicação também contará com um usuário que será o administrador do sistema. Caso a combinação do nome de usuário e senha inseridas não retornar nenhum paciente ou médico, o usuário receberá como resposta as páginas destinas ao administrador. O administrador tem acesso a todas as informações contidas na aplicação, excluir dados de consultas, médicos e pacientes livremente.

Tanto o paciente, tanto o médico pode procurar um ao outro e assim marcar a consulta, a consulta possui os dados dos dois envolvidos e quando criado, o médico pode editar essa consulta adicionando um diagnóstico ou receita para esse paciente.

Resumidamente, o programa é implementado seguindo o padrão MVC(Model, View, Control). As classes do pacote "socialdoc.model", implementam os conhecidos métodos *getters* e *setters,* que encapsulam os atributos das classes, para que não seja possível agentes não autorizados acessarem os atributos privados. Nesse pacote ocorre também a herança de "Usuario", anotado para ser uma herança *SINGLE\_TABLE,* dividindo a classe em "Medico" e "Paciente"*.*

No pacote "socialdoc.control" estão as classes *bean,* que controlam o funcionamento dos agentes medico, usuario e consulta, relacionando-os com os *views.* Nas classes dessa divisão estão métodos que recebem parâmetros do tipo medico, paciente, consulta, relacionando-os para fazer as operações com o banco de dados.

Além desses, existem pacotes como "socialdoc.filter" e "socialdoc.repository", cujos métodos dão conta de filtrar a sessão, para que um usuário que um usuário não receba outra sessão caso ele já esteja com uma alocada, fazendo isso através do método "doFilter", que recebe uma *request, response* e *chain,* processando as informações recebidas e, caso o usuário tenha uma sessão, será lançada uma exceção. Por fim, os "repositórios" cuidam especificamente do manuseio dos dados no banco referentes a usuários e consultas.

DIFICULDADES E O QUE NÃO FUNCIONOU

Houve dificuldades ao programar as principais funcionalidades do sistema, então demos o foco nas principais funcionalidades e outras implementações foram deixadas de lado, por isso temos um sistema entregue com não todas as funcionalidades prometidas, mas com muito do que foi aprendido na aula de DW2.

A grande dificuldade do início do desenvolvimento foi trabalhar com datas, mas essa foi uma das funcionalidades que foi o foco está tudo já implementado. A integração das páginas .xhtml co o Bootstrap também foi uma das dificuldades, mas todas as páginas estão estilizadas para serem responsivas, apenas houve um problema ao fazer o logout de uma página privada de médico ou paciente, a página index ficou sem o efeito da estilização, mas volta ao seu estado inicial se atualizada.

Devido ao tempo de entrega não foi feito a internacionalização do sistema e também elaborar com janelas modal (como um pop-up do Bootstrap) para os usuários visualizarem as receitas e observações de modo mais explícito em vez de ver o texto na tabela e também organizar em abas as funcionalidades do administrador. Mas o sistema está entregue com o mínimo de funcionalidades, como foco na funcionalidade principal de marcar e manter as consultas dos usuários.