



FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Departamento de Engenharia Informática
Licenciatura em Engenharia Informática
Mobilidade em Redes de Comunicação 2011/2012

Aplicação Android Serviços de Saúde da Universidade de Coimbra



Gonçalo Silva Pereira 2009111643
Igor Nelson Garrido da Cruz 2009111924

Índice

1. Introdução.....	2
2. Descrição das funções do sistema.....	3
3. Descrição da interacção Humano-Dispositivo electrónico (computador, smartphone) , com printscreens da GUI.....	4
4. Descrição da estrutura do código (classes e ficheiros):	
• Estrutura de código do cliente.....	7
• Estrutura de código do servidor.....	9
5. Armazenamento de informação no cliente e no servidor.....	
6. Explicação dos principais problemas e de como os mesmos foram resolvidos.....	12
7. Conclusões.....	13

1. Introdução

Este projecto tem por base a concepção de uma aplicação destinada à plataforma Android para aplicação e desenvolvimento dos conhecimentos adquiridos nas aulas de Mobilidade em Redes de Comunicação. Baseia-se numa arquitectura servidor-cliente, onde os clientes podem efectuar reservas, seleccionando a especialidade, o médico e o slot desejado e efectuando o pagamento da mesma através do sistema de créditos, ver as consultas reservadas, receber uma confirmação via SMS, enviar uma reserva via e-mail, armazenar a informação das reservas localmente.

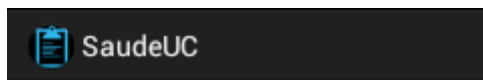
Para implementar as diversas funcionalidades propostas, tivemos que recorrer aos conhecimentos adquiridos nas aulas desta cadeira e de outras cadeiras de Licenciatura, como por exemplo Introdução as Redes de Comunicação, Protocolos de Comunicação e Sistemas Distribuídos, tais como a utilização de sockets TCP/IP, Threads.

2. Descrição das funções do sistema

As funções base são:

- Autentificação no sistema
- Navegar pela lista de especialidades, médicos e de consultas
- Seleccionar o time-slot desejado da lista dos time-slots disponíveis
- Fazer o pedido de uma reserva
- Depois da confirmação ser recebida , efectua o pagamento da consulta através de um sistema de créditos, a aplicação verifica se o cliente tem créditos para efectuar a reserva da consulta ou não,
- Receber a confirmação da reserva e do pagamento por SMS
- Guardar as informações das reservas localmente
- Consultar todas as reservas
- Enviar as reservas guardadas localmente para um e-mail

3. Descrição da interacção Humano-Dispositivo electrónico (computador, smartphone) , com printscreens da GUI



Assim que iniciamos a aplicação, é-nos mostrado um menu de Autentificação, onde podemos inserir os nossos dados.

Dados de Login

Username

Registrar Login Sair

SaudeUC

Registo de Utilizador

Username:

Password:

Confirm Password:

Confirmar Cancelar

No caso de não termos uma conta temos um botão para ir para o menu de registo onde temos de colocar o username pretendido, a password e a password de confirmação, o servidor verifica se o utilizador já existe ou não, se existir remete uma mensagem a dizer que está indisponível.



Creditos: 700

Especialidades

Consultas Marcadas

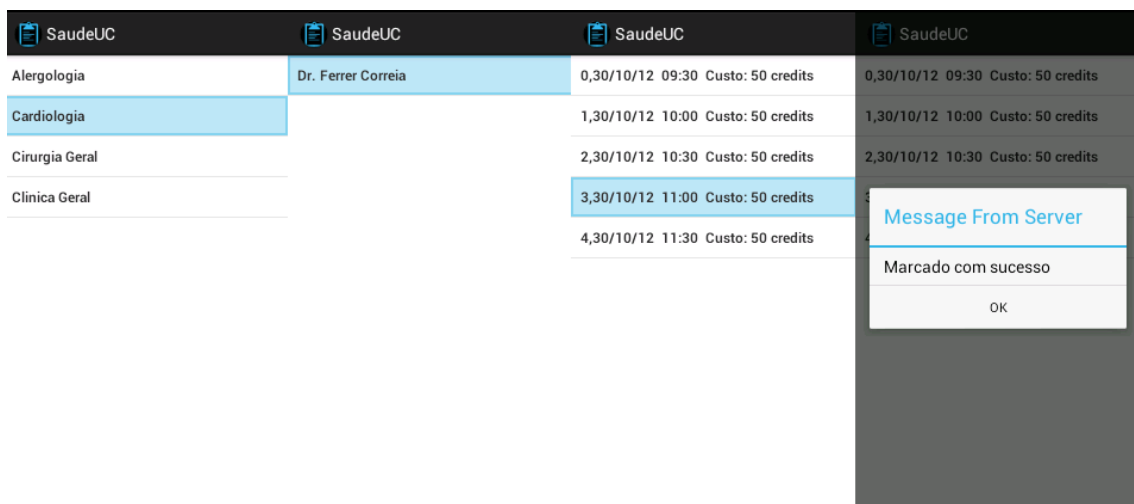
Comprar Creditos

Mapa


Sair

Após o processo de autenticação, entramos no menu principal da aplicação, onde podemos ver o nosso saldo de créditos na aplicação, podemos ver as especialidades onde podemos reservar consultas, consultar as consultas marcadas e enviar essa informação para o e-mail, comprar mais créditos, ver a localização dos serviços de Saúde da Universidade de Coimbra num mapa e, por fim, sair.

Para reservar uma consulta, temos de escolher a especialidade, escolher o médico, escolher o slot e enviar essa informação para o servidor, se o utilizador tiver créditos suficientes e se o slot estiver realmente livre, o utilizador recebe uma mensagem a dizer que foi marcado com sucesso.



Para consultar as reservas de consultas escolhemos a opção: “Consultas Marcadas”, onde nos vão ser mostradas as várias consultas que anteriormente reservamos. Temos também a possibilidade de enviar as informações das várias consultas por e-mail.

 SaudeUC

SaudeUC: 30/10/12 10:00
Medico:Dr Lino Chieira
Especialidade:Alergologia

Para comprar créditos de modo a termos permissão para efectuar reservas de consultas escolhemos a opção: “Comprar Créditos”, de seguida são nos mostrados vários campos que temos de preencher, como o nosso nome, sobrenome, o numero do cartão de credito, a validade do mesmo e o CSV. O servidor posteriormente vai validar estas informações e creditar os ditos créditos ao utilizador.

Enviar para Email

Anterior

Seguinte



Por fim podemos visualizar num mapa a localização exacta dos Serviços de Saúde da Universidade de Coimbra, assim como mostra o seguinte printscreen da aplicação

4. Descrição da estrutura do código (classes e ficheiros)

Estrutura de código do cliente:

Classes do cliente:

- Carregar.java

Nesta classe estão definidos os botões e as caixas de texto onde o utilizador insere os seus dados para carregar a sua conta com mais créditos, enviando o pedido de carregamento para o servidor, que vai verificar se os dados são válidos ou não.

- Consultas.java

Nesta classe estão definidos os botões e as caixas de texto do layout para possibilitar ao utilizador consultar as consultas reservadas e para enviar essa mesma informação para o e-mail.

- Data.java

Classe onde estão declarados atributos para receber os dados vindos do servidor, como por exemplo: o username, a password, um array de especialidades, um array de médicos, um array de slots, um array de reservas e um array de Id's.

- Doctors.java

Classe que representa os médicos.

- Menu.java

Classe onde estão definidos os botões e uma caixa de visualização de texto que constituem o menu principal. Onde é possível reservar uma consulta, ver as consultas já reservadas, comprar créditos, verificar os alarmes/notificações das próximas consultas, ver um mapa da localização dos Serviços de Saúde da Universidade de Coimbra.

- Register.java

Classe que gere a criação de contas, tendo o cliente de inserir os seus dados, username, password e password de confirmação.

- SaudeUC.java

Nesta classe estão definidos os vários botões e caixas de texto necessárias para o utilizador preencher os seus dados de autenticação no sistema.

No caso de o utilizador não ter conta para se autenticar no sistema, ele terá de se registar.

- Slots.java

Slots é a classe que armazena as informações relativas às consultas de cada médico.

- Specialities.java

Classe responsável por representar as especialidades médicas existentes no servidor.

Em relação às threads , o nosso cliente tem uma thread principal, e quando quer comunicar com o servidor cria outra thread só para fazer a comunicação, uma vez que a thread de comunicação não pode ser a mesma thread que trata do GUI, uma vez que se fosse possível, iria notar-se lentidão na aplicação e até mesmo crashar completamente a aplicação.

Estrutura de código do servidor:

O servidor está programado em Java e as suas classes principais são:

- Consulta.java

Uma consulta é definida por um id inteiro, um boolean que define se a mesma já está marcada ou não, o username do utilizador que a reservou, a data e a hora em que a mesma se vai realizar e o preço da mesma.

Temos também uma variável para guardar o ID da ultima consulta que terá de ser guardado em ficheiro para os dados ficarem consistentes.

- Doctor.java

Cada médico tem um nome e um ArrayList de Consultas, em que cada consulta tem os seus slots disponíveis e marcados.

- GenerateSchedule.java

Classe que adiciona os slots das consultas a cada médico e especialidade, neste momento todos esses dados estão explicitamente declarados nesta classe, para a disponibilização desta aplicação ao público em geral, é necessário guardar todos os dados das consultas em ficheiros.

- ServerMain.java:

Classe onde estão declarados explicitamente os dados das especialidades dos médicos e os dados dos utilizadores em ArrayLists.

Ficheiro onde está guardado o número da porta para a criação do socket e o numero máximo de conexões.

No main() desta classe são criadas as especialidades, que têm um ArrayList de médicos que por sua vez vai ter um ArrayList de consultas (especificado na classe Doctor.java).

Nesta classe estão também definidos dois fluxos de dados, um de entrada como DataInputStream e um de saída como PrintStream, estes fluxos servem para fazer a comunicação com o cliente nos dois sentidos: Cliente -> Servidor e Servidor -> Cliente.

Estão definidas também os vários tipos de mensagens String que são recebidas e as mensagens que vão ser enviadas/imprimidas da parte do cliente.

- Speciality.java

Classe que possui o nome da especialidade e o ArrayList de médicos dessa mesma especialidade.

- User.java

Classe que tem como atributos o username, a password, o numero de créditos actual do utilizador e um boolean para dizer se o mesmo está Online ou não.

A nível de threads, o nosso servidor tem uma thread principal, uma thread para cada cliente que se liga e uma thread para gerir os slots das consultas.

A thread que gera os slots das consultas tem como objectivo criar os vários slots das consultas para mostrar ao cliente assim que o mesmo entrar na aplicação. Neste momento estão inseridos os para uma semana no servidor, pois é suposto esta thread criar os dados dos slots no inicio da semana, estando a mesma a dormir o resto do tempo.

5. Armazenamento de informação no cliente e no servidor

É necessário armazenar alguma informação localmente no cliente, para o cliente aceder mais facilmente à mesma e provar que o slot foi reservado no dia da consulta, caso exista alguma falha no sistema e/ou no servidor, o cliente poder provar que foi efectuada a reserva do slot.

No servidor as informações são explicitamente adicionadas nas classes, mas para garantir a fiabilidade e a consistência dos dados no mesmo no momento da disponibilização da aplicação ao universo da Universidade de Coimbra terá de ter uma Base de Dados, assim como um servidor de Backup, para recuperar as ligações no caso do servidor principal falhar por alguma razão.

6. Explicação dos principais problemas e de como os mesmos foram resolvidos

Em termos de comunicação com o servidor, começámos por utilizar uma arquitectura em que existia uma thread permanentemente em execução responsável por comunicar com o servidor. No entanto, ao longo do desenvolvimento da aplicação, verificámos que era muito complicado utilizar esta estrutura, pois o Android não permite que a thread responsável pelo GUI faça comunicação “Network”.

Deste modo a solução pela qual optámos, passa por ter uma thread auxiliar que é lançada quando o cliente quer comunicar com o servidor.

Na programação e teste da aplicação com uma Virtual Machine do Android no computador, verificamos que o mesmo é lento e que não tem nenhum gestor de ficheiros para verificar se a escrita em ficheiro dentro do dispositivo do cliente é feita ou não.

Tivemos vários problemas na integração de um mapa através do Google Maps, para visualizar a localização dos Serviços de Saúde da Universidade de Coimbra, uma vez que por vezes quando abrimos os mapas apenas nos aparece a grelha faltando os mapas em si. Conseguimos ultrapassar este problema adicionando os mapas de outra forma, possivelmente mais correcta.

7. Conclusão

O desenvolvimento de aplicações para dispositivos móveis apresenta um papel muito importante no paradigma actual das aplicações distribuídas. É função do engenheiro informático criar novas funcionalidades e comodidades ao utilizador e melhorar a qualidade dos serviços já existentes.

Deste modo, é importante aprender a trabalhar com ferramentas novas como é o caso da plataforma Android.

Através do trabalho realizado foi fácil averiguar que esta plataforma é capaz de responder bem aos requisitos necessários, e após alguns problemas, a elevada quantidade de informação consegue auxiliar o desenvolvedor, possibilitando uma aprendizagem bastante rápida.

Com este projecto utilizámos não só conceitos aprendidos na cadeira de Mobilidades em Redes de Comunicação, mas também de outras cadeiras como Sistemas Distribuídos.

Aprendemos como trabalhar com a interface gráfica do Android, como comunicar com um servidor responsável por fornecer serviços e como estruturar software distribuído.