Universidade do Porto
Faculdade de Engenharia
FEUP

Relatório de
Computação Móvel

APOSTAS DESPORTIVAS EM ANDROID:

Porto, Novembro de 2010

Turma: 5MIEIC1

Grupo:

Nº 060509116	João Xavier
Nº 060509029	Paulo Pinto

APOSTAS DESPORTIVAS EM ANDROID:

Porto, Novembro de 2010

Ano: 5º

Semestre: 1º
Turma: 1

Nº 060509116	João Cristóvão Afonso Sampaio Xavier	ei06116@fe.up.pt
Nº 060509029	Paulo André Teixeira Pinto	ei06029@fe.up.pt

Docentes:

Doutor Eng.º António Miguel Pontes Pimenta Monteiro

Relatório realizado no âmbito do projecto da disciplina de Computação Móvel do 5º Ano do 1º Semestre do Mestrado Integrado em Engenharia Informática e Computação da Universidade do Porto.

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Declaração de originalidade

Os autores declaram que o relatório e o código fonte submetido são da sua autoria, excepto nas partes explicitamente assinaladas com referência a respectiva fonte.

João Cristóvão Afonso Sampaio Xavier
Paulo André Teixeira Pinto

Resumo

Este relatório incide sobre a elaboração e documentação de uma aplicação cliente-servidor com o objectivo de gerir apostas em torneios de futebol. A aplicação, desenvolvida para terminais *Android*, permite de forma intuitiva efectuar apostas sobre jogos e consultar estatísticas dos torneios e equipas.

Entre as tecnologias utilizadas destacam-se as seguintes: *web services* (RESTFul), JSON, Jersey e a *framework* de desenvolvimento para *Android*.

ÍNDICE

Lista de Figuras

Lista de Tabelas

1.Introdução

1.1 Enquadramento

O trabalho descrito neste relatório é uma aplicação em *Android* e foi desenvolvido no âmbito da disciplina de Computação Móvel, no 5º ano do curso de Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

1.2 Motivação

A principal motivação para a realização deste projecto é a iniciação numa área da informática até agora pouco explorada – desenvolvimento de aplicações para dispositivos móveis.

O *Android* é um sistema operativo que é corrido sobre o núcleo Linux e desenvolvido pela *Google*. As bibliotecas disponíveis permitem aos programadores desenvolver aplicações em *Java*, interagindo desta forma com o dispositivo. Actualmente, este sistema é utilizado maioritariamente em telemóveis.

Em relação ao projecto, permitiu desenvolver competências e conhecimentos de desenvolvimento de aplicações para dispositivos móveis, mais concretamente para *Android*, tornando-se uma mais-valia no percurso académico. É importante realçar que os elementos do grupo apresentam um grande interesse por aplicações com foco nesta tecnologia.

1.3 Objectivos

O objectivo principal do trabalho é elaborar e documentar uma aplicação para um dispositivo Android, utilizando a framework disponível pela *Google*. Para aprofundar os conhecimentos sobre este sistema operativo foi elaborada uma aplicação de apostas em jogos. Esta aplicação permite, de forma

intuitiva, efectuar apostas sobre jogos de futebol, destinado a pequenos torneios. A aplicação deve permitir a gestão de utilizadores que têm associado, além da informação pessoal, um crédito que lhes permite apostar em jogos. Os dados deverão ser previamente inseridos na base de dados do servidor.

1.4 Estrutura do relatório

Este trabalho encontra-se estruturado em cinco capítulos.

O primeiro é composto por esta introdução ao trabalho.

O segundo capítulo corresponde à descrição do projecto, onde são mencionados os requisitos e restrições.

No capítulo seguinte - implementação - é apresentada a arquitectura da aplicação desenvolvida.

O quarto capítulo apresenta uma sequência de transições de ecrãs na utilização da aplicação.

Por fim, no último capítulo é apresentada a conclusão do trabalho.

2. Descrição do Problema

Este projecto tem como objectivo o desenvolvimento de uma aplicação para dispositivos Android que permita realizar apostas de forma fácil e cómoda por diversos utilizadores. O sistema foca-se em apostas em jogos de futebol, integrados em pequenos torneios de duas mãos, envolvendo 4 ou 5 equipas.

A aplicação deve ser capaz de gerir dois tipos de utilizadores distintos: administrador e utilizador comum. Os utilizadores deverão ter sido previamente registados no sistema, apresentando um crédito que lhes permite apostar em jogos.

Os utilizadores autenticados no sistema têm a possibilidade de efectuar apostas sobre jogos que ainda não foram realizados; a interface no dispositivo vai permitir a selecção dos jogos de forma agradável para o utilizador. Ao apostar pode-se definir o montante desejado (logo que não ultrapasse o crédito disponível). Caso a aposta seja certa os pagamentos efectuados serão os seguintes:

- 120% do valor apostado para apostas em vitória e derrota;
- 150% do valor apostado para empate;
- 300% do valor apostado com o resultado certo.

Esta aplicação permite obter um panorama geral do torneio; pode-se consultar estatísticas relativas a jogos realizados e às equipas intervenientes (posição, pontuação, numero de vitórias, empates e derrotas, golos marcados e sofridos, estatísticas em casa e fora de casa, resultados concretos de jogos, etc).

O administrador tem a seu cargo duas funções:

- Indicar o resultado de um jogo realizado;
- Desencadear o crédito aos apostadores que acertaram.

A interface deve ser intuitiva permitindo uma interacção agradável ao utilizador do dispositivo móvel.

3. Implementação

3.1 Arquitectura do Programa

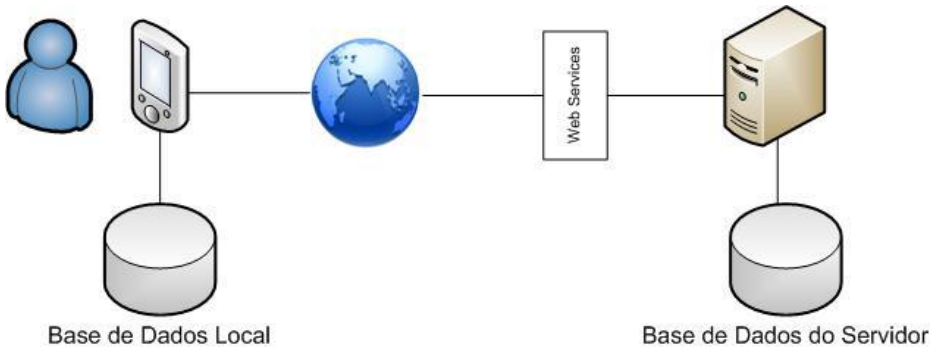


Figura 1 - Arquitectura da aplicação

A aplicação desenvolvida pode classificar-se como cliente-servidor. O servidor tem a responsabilidade de gerir os vários pedidos efectuados pelos clientes e guardar a informação relativa aos torneios (equipas intervenientes, jogos, calendarização, etc). O cliente guarda a informação pertinente para o utilizador que está autenticado no sistema, permitindo um acesso mais rápido aos dados e independente do servidor. O servidor e cliente encontram-se sincronizados de forma a que o cliente não apresente informação desactualizada. A comunicação é realizada através de *web services* (RESTful) a correr no servidor remoto.

As bases de dados são quase gémeas, tendo a mesma estrutura, de modo a que a troca de dados entre o cliente e o servidor seja mais eficiente e intuitiva. De seguida, é apresentado o modelo relacional da base de dados criada – Figura 2.

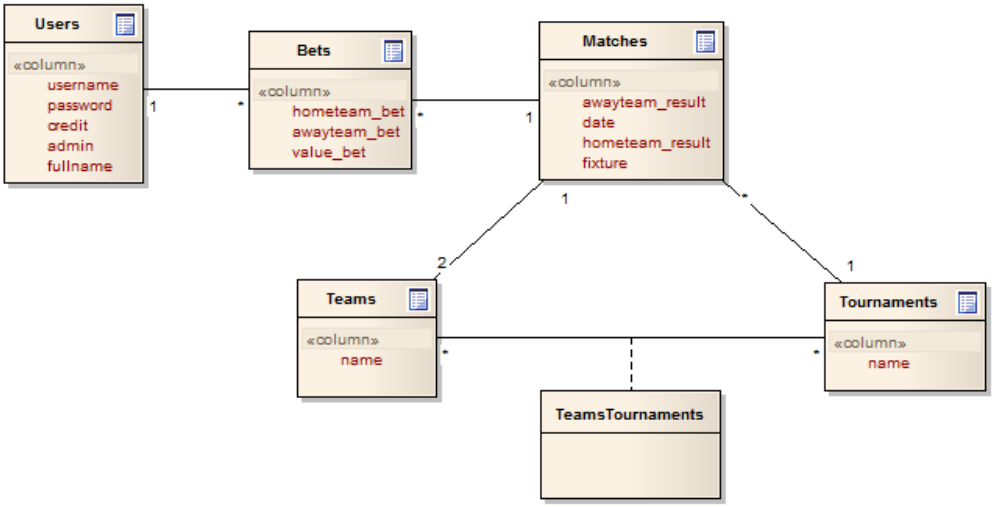


Figura 2 - Modelo Relacional da base de dados

Este esquema relacional permite o desenvolvimento de uma base de dados que suporta as diferentes funcionalidades da aplicação. A arquitectura desenvolvida permite estender facilmente os requisitos especificados no enunciado.

As bases de dados foram implementadas em SQLite, um sistema de gestão de base de dados direccionado para aplicações simples em termos de administração, implementação e manutenção.

3.1.1 Servidor

Estrutura

A arquitectura do cliente divide-se estruturalmente em 4 *packages*:

- wsbfail.tables;
- wsbfail.resource;
- wsbfail.service;
- wsbfail.syncdb.

wsbfail.tables

Esta secção apresenta as classes encarregues de modelar e manipular os objectos da base de dados: utilizadores, apostas, torneios, etc, apresentando funcionalidades básicas: get/set.

wsbfail.resource

Este package apresenta as classes que implementam os *web services* disponíveis. Para mais detalhes consultar a secção 3.1.2 Web Services.

wsbfail.service

Esta secção permite guardar em cache informações relevantes para o decorrer da aplicação, permitindo acesso à informação sem consultar a base de dados, sendo mais eficiente do que a consulta aos dados persistentes (base de dados remota). Desta forma, estão implementadas várias funções responsáveis pelo tratamento e manipulação deste conjunto de dados.

wsbfail.syncdb

Este package é responsável por:

- estabelecer a conexão entre a base de dados remota e o servidor;
- criar a base de dados;
- manter a informação das bases de dados dos clientes sincronizada com a base de dados remota.

No momento em que são efectuadas actualizações na base de dados, o servidor armazena um histórico de operações realizadas e associa um número de versão da base de dados. O cliente, quando necessita de actualizar a base de dados, envia a versão em que se encontra a sua base de dados e o servidor encarrega-se de ler o histórico das operações e enviar as actualizações necessárias. Assim, a informação a transmitir é reduzida, sendo uma mais-valia em dispositivos deste tipo, onde os dados transmitidos através da Internet apresentam um custo elevado.

Tecnologias

JDBCdrive

Java Database Connectivity ou JDBC é um conjunto de classes e interfaces escritas em Java responsáveis pelo envio de instruções SQL para uma base de dados relacional.

Foram implementadas transacções nos acessos à base de dados, de forma a não a corromper por alguma falha de *software*/*hardware*. A concorrência de acessos a base de dados também é garantida.

Glassfish

GlassFish é o servidor *open source* utilizado neste projecto. Foi desenvolvido pela Sun Microsystems para a plataforma Java Enterprise Editions.

Apache Tomcat

O Tomcat é um servidor web Java (*servlet container*). Este servidor permitiu a implementação puramente em Java de serviços web HTTP. Inclui ferramentas de configuração e gestão, que podem ser editados manualmente a partir de ficheiros formatados em XML.

JavaServer Pages

JSP é uma tecnologia utilizada para o desenvolvimento de aplicações Web (em semelhança a ASP.net e PHP). Visto ser baseado em Java, permite a execução em diversos sistemas operativos. O programador pode desenvolver aplicações para aceder à base de dados, manipular arquivos de texto, capturar informação a partir de formulários, etc. Uma página criada com a tecnologia JSP após instalada num servidor é transformada num *servlet*.

3.1.2. Web Services

REST (Representational State Transfer) é uma técnica de engenharia de software para sistemas hipermédia distribuídos como a web. Está relacionado directamente com um conjunto de princípios de arquitectura, sendo utilizado actualmente para descrever qualquer interface web simples, sem abstracções adicionais dos protocolos baseados em padrões de trocas de mensagens.

Princípios:

- protocolo cliente/servidor sem estado (cada mensagem HTTP contém toda a informação necessária para compreender o pedido);
- conjunto de operações bem definidas HTTP: POST, GET, PUT e DELETE.
- sintaxe universal, sendo cada recurso unicamente direccionado através do URI.

Figura 3 - Operações HTTP disponíveis

Os formatos XML e JSON estão disponíveis no Jersey (JAX-RS: Java API for RESTful Web Services). Uma das principais vantagens no uso da JSR-311/Jersey é eliminar a validação de pedidos e o mapeamento dos mesmos em classes e métodos que devem processá-las.

Neste projecto foi utilizado JSON para trocas de mensagem entre o servidor e o cliente.

Lista de *Web services*

UserResource

Todos os *web services* relativos aos utilizadores do sistema encontram-se especificados na classe UserResource.java. Esta classe engloba funcionalidades essenciais para a execução da aplicação:

- estabelecer a ligação entre o servidor e o cliente;
- autenticação dos utilizadores;
- obter a informação do utilizador;
- obter apostas realizadas, realizar novas apostas e atribuir crédito aos utilizadores;
- atribuir resultados a jogos.

De seguida, apresenta-se a lista de *web services* implementadas nesta classe:

Tabela 1 – *Web services* da classe UserResource

Tipo de pedido	URL	Descrição
@GET	http://{ip}:8080/WSBFail/rest/user	Verifica se o servidor está *online* e pronto para iniciar conexão com o cliente. Retorna o código HTTP 200 em caso de disponibilidade e 204 em caso contrário. É o único *web service* disponível caso não esteja autenticado no sistema.
@GET	http://{ip}:8080/WSBFail/rest/user/username:{username}&password:{password}	O utilizador, para ter acesso às outras funcionalidades do sistema, tem necessariamente que realizar login. Este *web service* é disponibilizado para esse fim. É passado por parâmetro o nome de utilizador e a palavra passe codificada em MD5 para maior segurança na autenticação. Após uma autenticação com sucesso, o utilizador fica ligado ao sistema e é possivel aceder aos outros *web services*. Devolve o código HTTP 401 caso o conjunto utilizador/password esteja incorrecto e 202 se as credenciais forem aceites.

@GET	http://{ip}:8080/WSBFail/rest/user/username:{username}	*Web service* para fazer logout do sistema. Retorna o código HTTP 404 caso o utilizador não se encontre registado no sistema ou não tenha realizado login e 200 em sucesso.
@GET	http:/{{ip}:8080/WSBFail/rest/user/{username}	Devolve a informação do utilizador.
@POST	http://{ip}:8080/WSBFail/rest/user/{username}/bet	Permite ao utilizador do sistema apostar em jogos que ainda não foram realizados, passando no URL o nome de utilizador e enviando a aposta que deseja realizar.
@GET	http://{ip}:8080/WSBFail/rest/user/{username}/bets	Devolve todas as apostas realizadas pelo utilizador especificado no URL.
@PUT	http://{ip}:8080/WSBFail/rest/user/{username}/match_result	Este *web service* só está disponível aos administradores do sistema e permite atribuir o resultado a um jogo que foi realizado. Devolve o código HTTP 200 em caso de sucesso e 401 em insucesso (o utilizador não está registado ou não é administrador do sistema).
@GET	http://{ip}:8080/WSBFail/rest/user/{username}/match/{match_id}/resolve_credit	Após a realização dos jogos o administrador tem a opção de realizar os pagamentos aos apostadores.

BigPushResource

Apresenta *web services* para requisitar a informação pública da base de dados quando o cliente nunca esteve conectado ao sistema ou por algum motivo não tem os dados na base de dados local. São disponíveis 4 serviços.

- @GET http://{ip}:8080/WSBFail/rest/bigpush/matches
- @GET http://{ip}:8080/WSBFail/rest/bigpush/tournaments
- @GET http://{ip}:8080/WSBFail/rest/bigpush/teams
- @GET http://{ip}:8080/WSBFail/rest/bigpush/tournamentsteams

Estes serviços devolvem a totalidade da tabela da base de dados remota, respectivamente a tabela Matches, Tournaments, Teams e TournamentsTeams. Todos os outros dados são confidenciais e cada utilizador só tem acesso aos seus dados.

SynchronismDBResource

As bases de dados local e remota têm obrigatoriamente de sincronizar a informação que é alterada, para este fim foi desenvolvido o seguinte *web service*:

- @GET http://{ip}:8080/WSBFail/rest/sync/username:{username}&version:{nVers}

Visto serem transmitidas informações confidenciais sobre os utilizadores, só um utilizador registado no sistema pode aceder ao *web service*. O cliente apresenta uma versão da base de dados local, que é enviada ao servidor para este determinar quais são as utilizações que se encontram em falta, enviando essa informação ao cliente.

3.1.3 Cliente

Estrutura

A arquitectura do cliente, a nível de código, divide-se estruturalmente em 6 packages:

- bfail.canvas
- bfail.db
- bfail.misc
- bfail.picker
- bfail.src
- bfail.ws

bfail.canvas

Secção que está encarregue de desenhar no ecrã através do sistema de desenho 2D da plataforma *Android*, o Canvas. É usado para representar os gráficos de informação da forma e posição das equipas.

bfail.db

Secção que interage directamente com a base de dados, tirando partido da interface *SQLiteOpenHelper* providenciada pela framework, designada para facilitar a abertura e actualizações na base de dados. Este package fornece um nível de abstracção adicional para gerar as diferentes vistas da aplicação através do uso de classes que estendem *SimpleCursorAdapter*, usadas para customizar a população das listas. Contém também uma classe que se encarrega de todas as *queries*, operações de inserção, modificação e remoção da base de dados.

bfail.misc

Este package contém classes de utilidade genérica, como constantes referentes a nomes de ficheiros, codificação de strings em MD5, informação sobre equipas e ainda uma classe abstracta usada para sincronização automática.

bfail.picker

Widget interna retirada do código fonte da plataforma *Android* que não está disponível directamente na *framework*. É usada para especificar resultados e apostas.

bfail.src

Todas as vistas (excluindo diálogos) estão incluídas neste package e estendem a classe *Activity*. Como todas utilizam a interface de base de dados, para evitar acessos concorrentes em actividades diferentes, no evento *OnPause* (mudança de actividade) a base de dados é fechada para posteriormente ser aberta no evento *OnResume*, chamado logo a seguir do evento *OnCreate* doutra actividade.

bfail.ws

Secção que interage com os *web services* e implementa a lógica da conexão a nível do cliente. Contém códigos customizados de conexão para *debugging* eficiente da aplicação e personalização de mensagens a transmitir ao utilizador. Inclui as vistas dos diálogos que involvem ligação ao servidor: fazer aposta, atribuir resultado a um jogo e o ecrã de autenticação.

Vistas

Devido à possibilidade que o *Android* faculta em especificação de *layouts* por XML, optou-se por separar ao máximo todas as vistas e estilos do restante código, identificando-se assim ficheiros contendo informações sobre cores, temas, *strings* e *layouts*.

Tendo em conta a possibilidade de visualização em *landscape*, teve-se o cuidado de garantir que toda a aplicação pode ser usada em qualquer modo, pela adaptação do *layout* de algumas vistas.

Tentou-se usar o maior número de funcionalidades diferentes a nível de interface, usando classes como *ListActivity*, *TabActivity*, *PreferenceActivity*, *Dialog*, *AlertDialog* para as janelas e *widgets* como o *NumberPicker* para tornar a atribuição de resultados e realização de apostas mais intuitiva.

Navegação inter-actividade

Uma actividade, em *Android*, é uma interface que toma controlo do primeiro plano do ecrã, que o utilizador visualiza. É guardada uma pilha de actividades para manter a sequência em que o utilizador as visitou, preservando desta forma uma navegação fluida e intuitiva.

De modo a simular um sistema multi-tarefas, as actividades têm o seguinte ciclo de vida (simplificado):

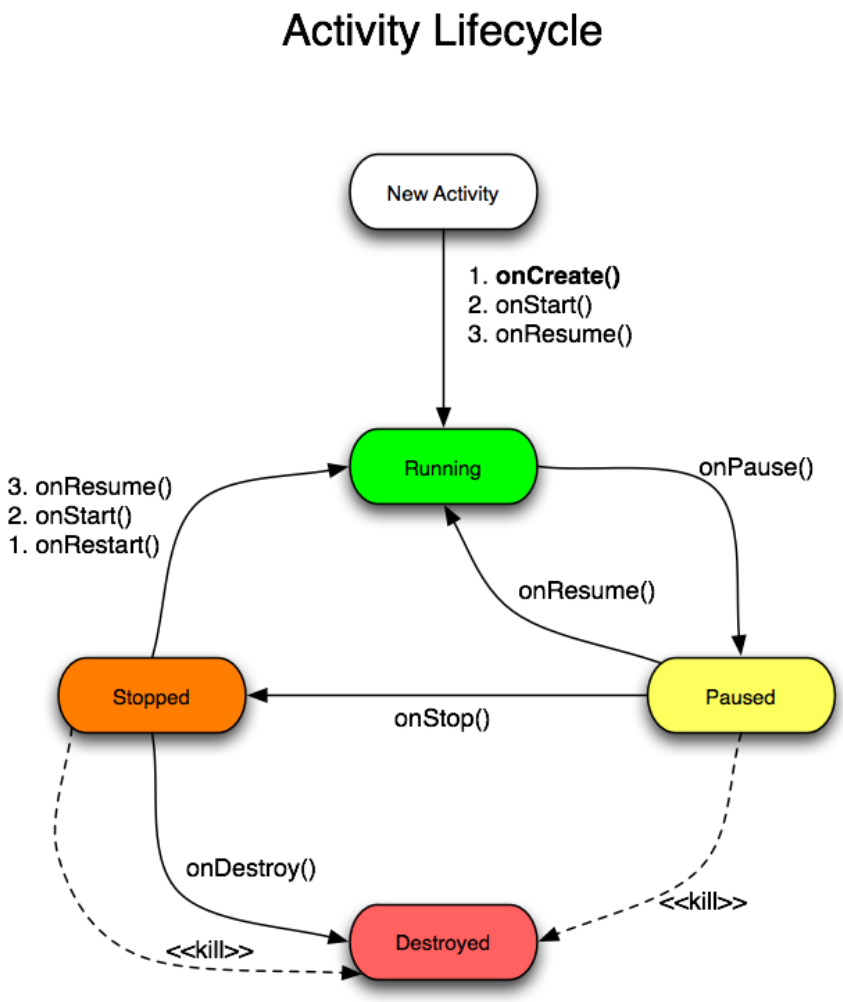


Figura 4 –Ciclo de vida das actividades em *Android*

A aplicação toma total partido dos eventos que são chamados pela mudança de estados da actividade.

Quando *onCreate()* é chamado, são feitas as inicializações que vão ser imutáveis no decorrer da actividade, isto é, não vão ser modificadas externamente enquanto esta estiver em primeiro plano.

onResume() é usado para carregar a informação passível de ser alterada, como vistas da base de dados. Desta forma garante-se que a cada vez que a actividade restaurar o primeiro plano, a informação é actualizada.

O evento *onPause()* faz a operação contrária ao evento anterior, isto é, guarda a informação da execução de modo a que a próxima actividade seja executada sem problemas.

onDestroy() é usado somente para libertar recursos que ficaram guardados em *cache* durante a execução do programa.

A única excepção no programa à regra da pilha de actividades é a transição entre a vista de jogos por data (jornada) e a tabela classificativa, já que a navegação entre estas é feita directamente e pode ser tão frequente que se optou por só manter uma instância de cada uma destas actividades na pilha.

3.2 Lógica da Ligação

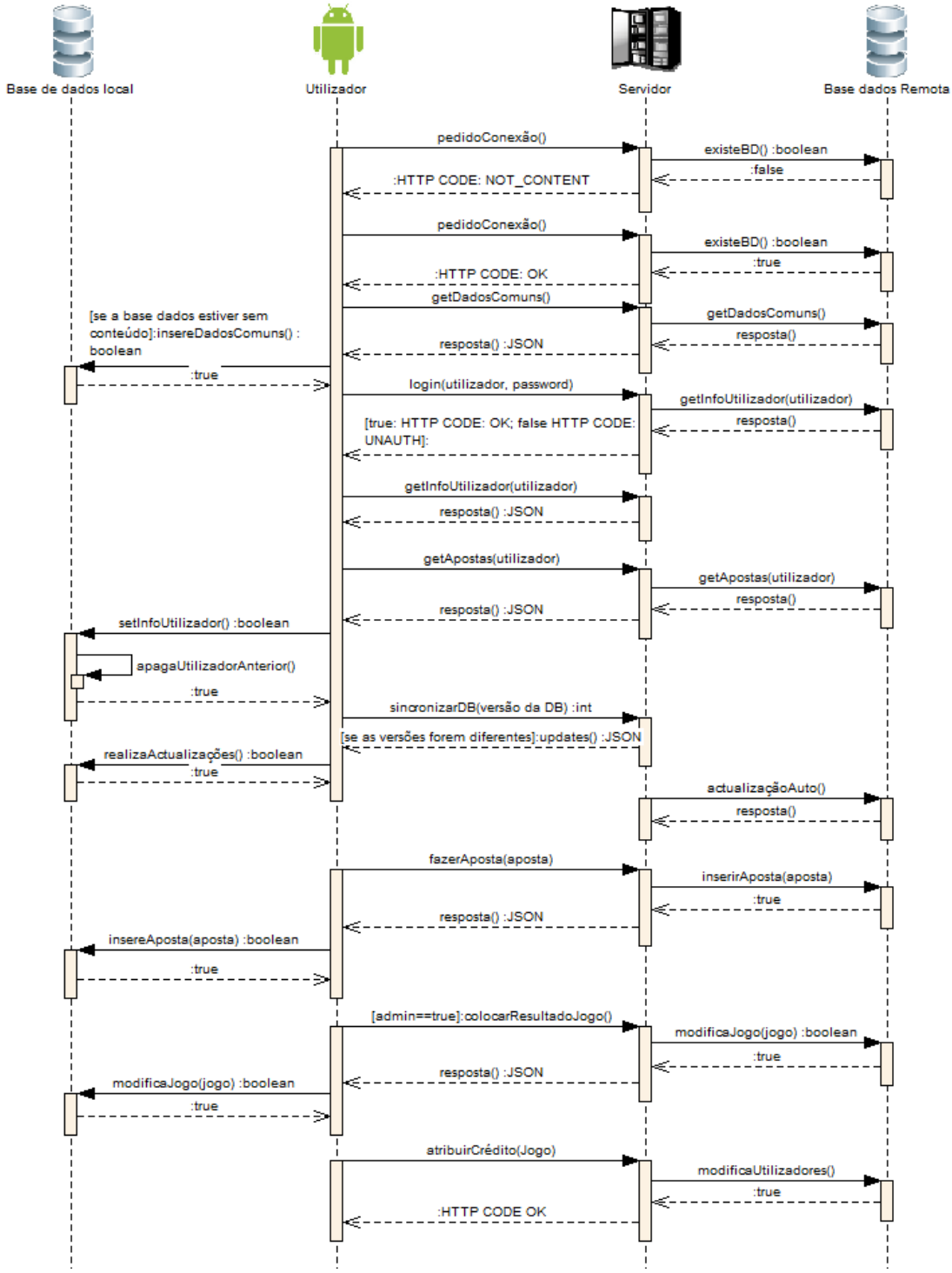


Figura 5 - Diagrama de Sequência: lógica da conexão

3.3 Funcionalidades

A aplicação foi implementada de uma forma mais genérica do que a especificada na descrição do trabalho.

Suporta um número variável de torneios e de equipas por torneio, tem um ecrã de conexão que informa o utilizador de todos os detalhes da ligação (bem sucedida, URL não encontrado e erros de autenticação).

Permite a consulta de informação do utilizador e das suas apostas, fornecendo um código de cores para classificar o resultado da sua aposta: verde se acertou no resultado, amarelo se o resultado estiver parcialmente correcto e vermelho se tiver falhado.

A palavra-passe é codificada em MD5 para aumentar a segurança da ligação.

Através dum menu de preferências é possível configurar se a informação do servidor, nome do utilizador e palavra-passe são guardadas entre diversas execuções da aplicação, se é feito um login automático sempre que possível e também se existe uma sincronização automática com o servidor a cada 30 segundos.

Todas as vistas que mostram informações de equipas e resultados têm opção de sincronização manual, com a excepção dos gráficos.

Um utilizador não autenticado tem acesso total à informação contida na base de dados local, referente aos torneios, equipas e jogos.

A tabela de classificações mostra as equipas ordenadas pela pontuação, tendo como critério de desempate a diferença de golos e, como critério de desempate adicional, o número de golos marcados. Na tabela de classificações é possível utilizar filtros de jogos em casa ou fora. Através dum clique numa equipa na tabela, surge a hipótese de saber mais sobre a equipa.

Na vista de detalhes de equipa, é possível visualizar os jogos que a equipa já realizou e os que ainda estão por realizar. Pode-se também consultar um gráfico de forma (historial de vitórias, empates ou derrotas) e da posição da equipa no decorrer das várias jornadas do torneio.

Se o utilizador se autenticar no sistema, na vista dos próximos jogos da equipa ou na vista dos jogos por jornadas, o utilizador pode abrir um menu

de contexto em determinado jogo para fazer uma aposta. Este ecrã só é mostrado se o jogo não tiver sido realizado, comunicando ao utilizador que já o foi em caso contrário. Não é permitido ao utilizador fazer apostas superior ao crédito que tem, e uma mensagem será apresentada se o tentar fazer. No caso de haver algum erro na conexão depois de tentar fazer a aposta, será mostrada uma mensagem a comunicar o sucedido.

Se o administrador se autenticar, poderá, seleccionando qualquer jogo, abrir um menu de contexto onde pode escolher atribuir o resultado a um jogo (independentemente de se ter realizado ou não) e desencadear o crédito das apostas nesse mesmo jogo, surgindo um diálogo para confirmação. Um ecrã de erro será mostrado no caso de haver problemas de conexão.

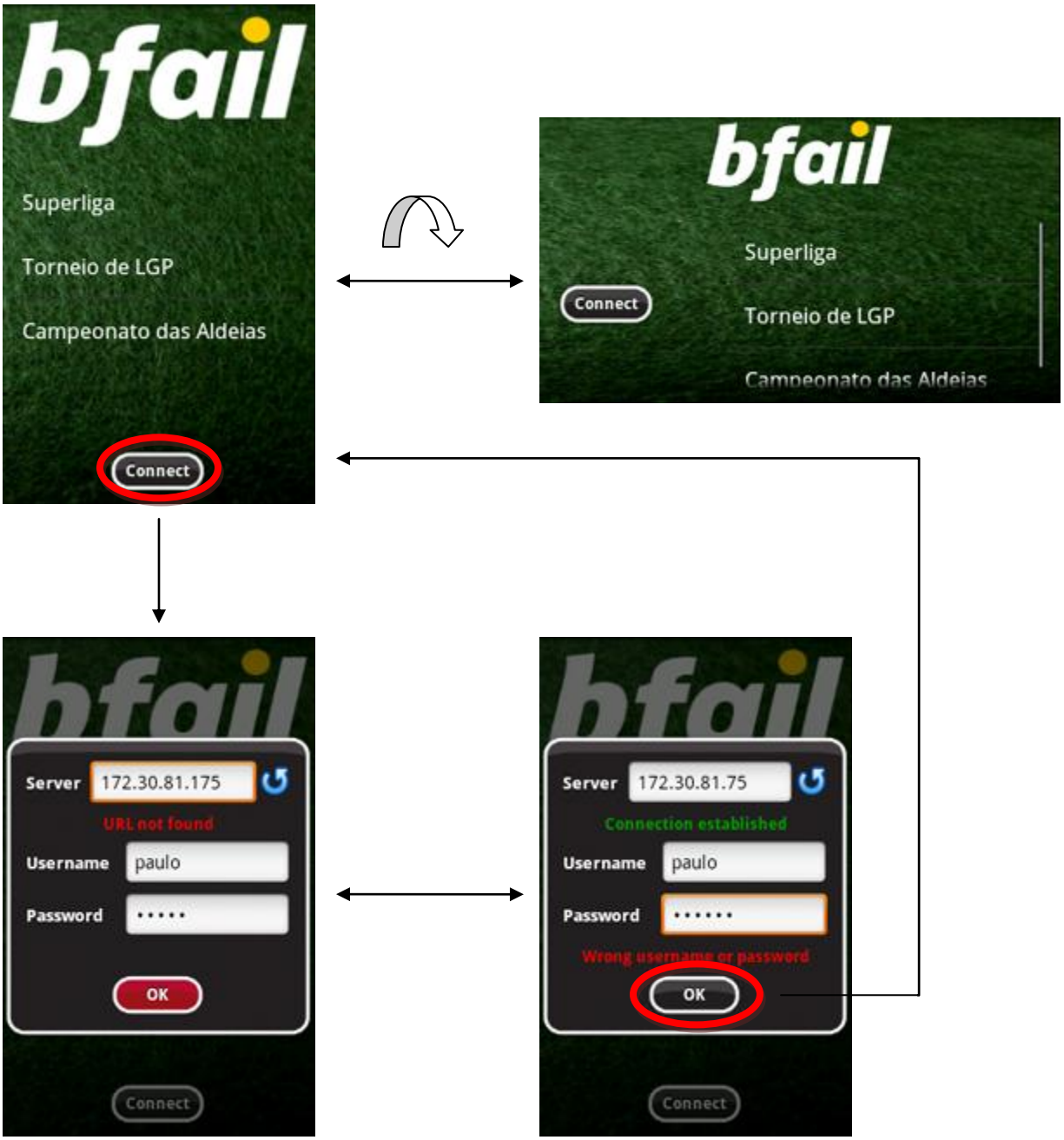
Como não existe restrição para o número de vezes que o administrador pode atribuir o resultado a um jogo ou desencadear o crédito, é esperado que não modifique resultados a jogos que já decorreram, a não ser em caso de engano, nem que atribua o dinheiro das apostas mais que uma vez.

3.4 Ambiente de Desenvolvimento

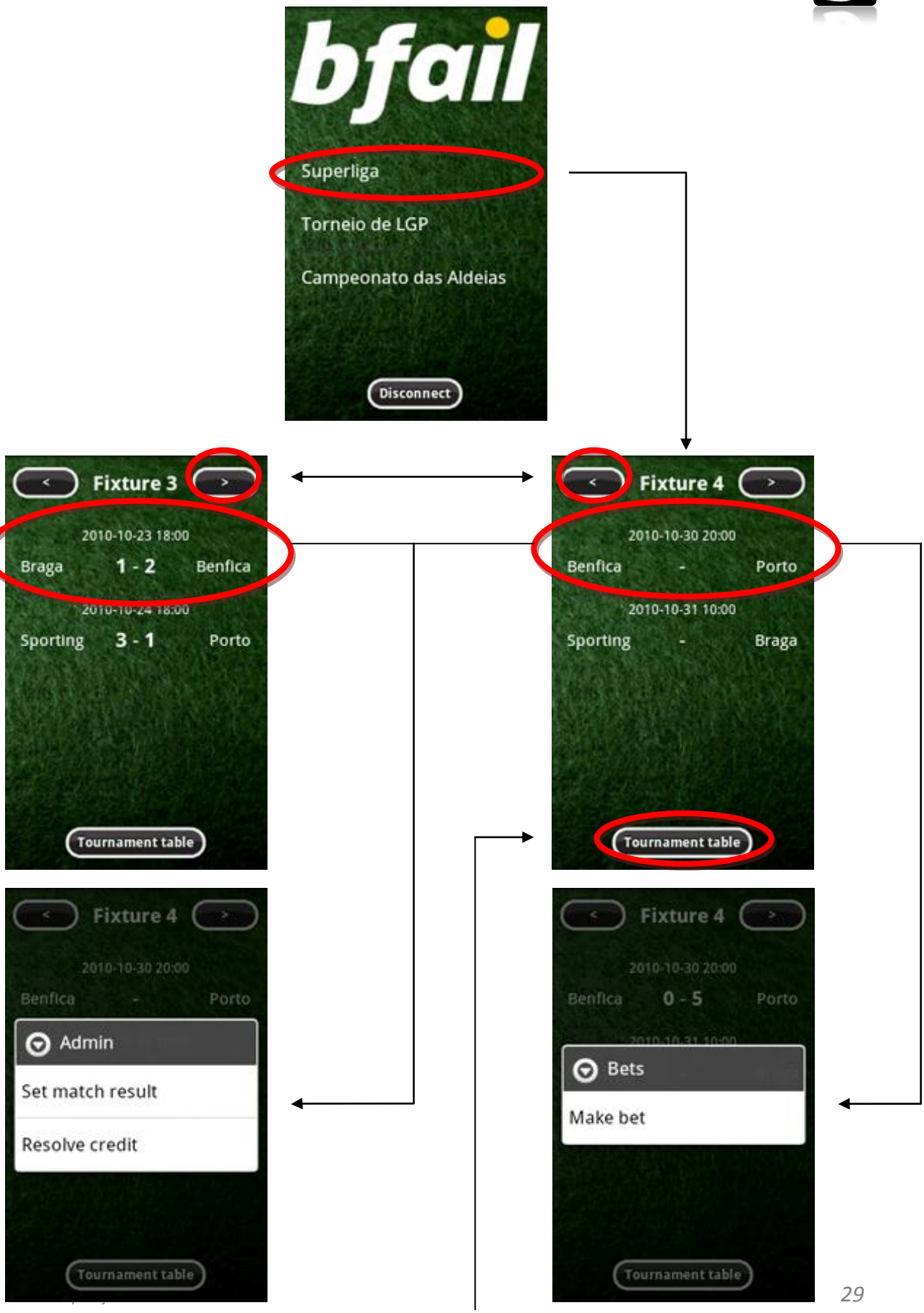
Os IDEs utilizados para desenvolvimento da aplicação foram o NetBeans IDE 6.9.1 para o servidor e o Eclipse Galileo para o cliente.

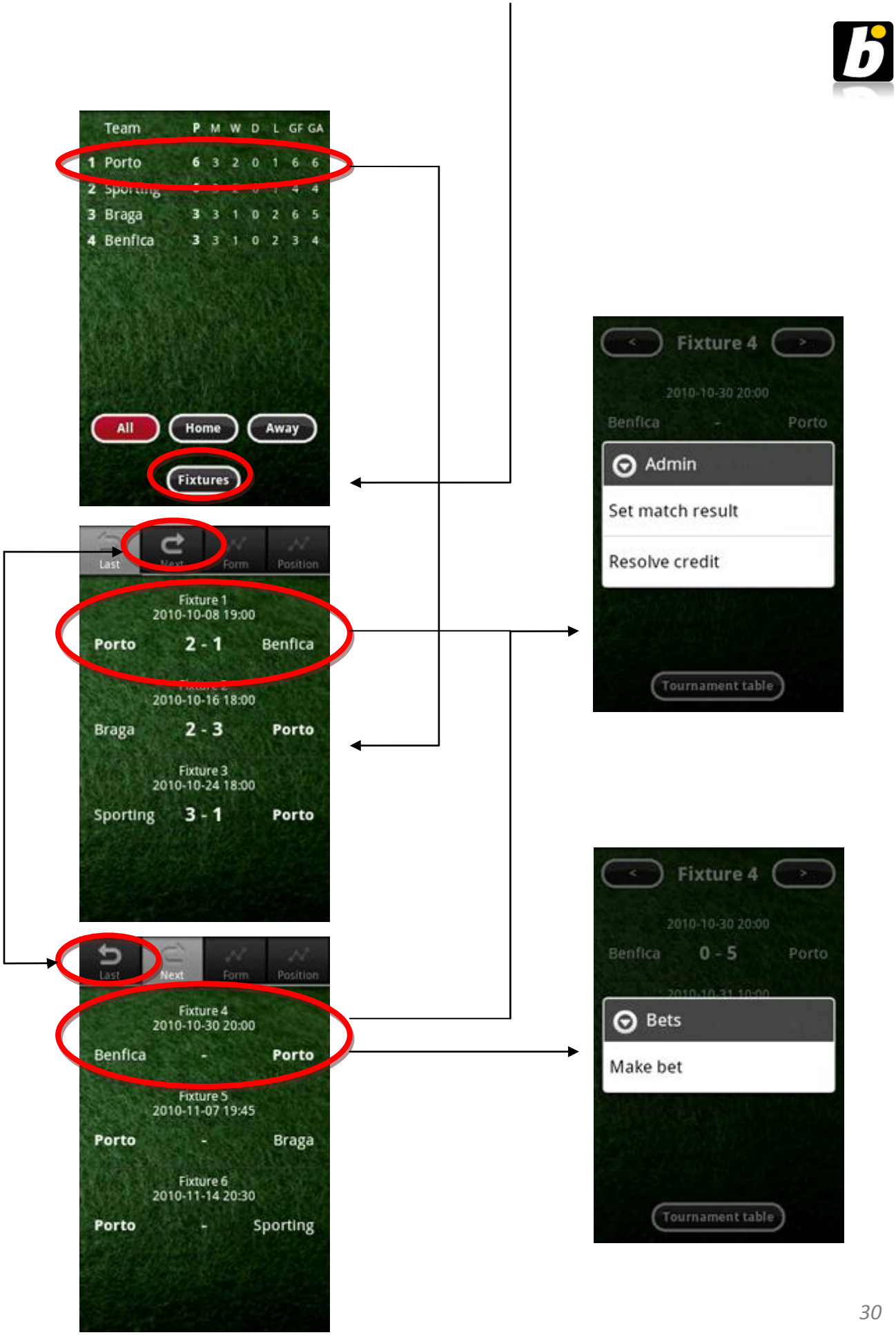
Para teste da aplicação *Android*, usou-se o telemóvel Samsung Galaxy 3 a correr o sistema operativo Android 2.1 e um *Virtual Device* a simular este mesmo telemóvel, com uma resolução de 240x400, um cartão de memória de 512 MB e uma densidade de ecrã LCD de 120.

4. Interface com utilizador

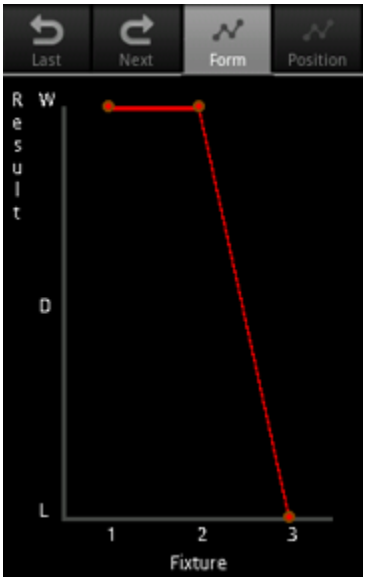


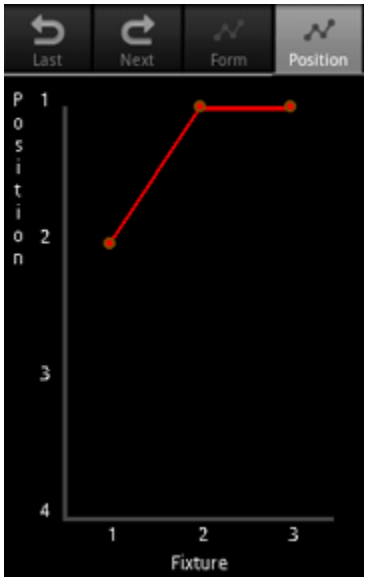

bfail
Superliga
Torneio de LGP
Campeonato das Aldeias
Disconnect
bfail
Superliga
Torneio de LGP
Campeonato das Aldeias
Profile
Bets
Synchronize
Preferences
Information
bfail
About bfail
Developed by:
João Xavier and Paulo Pinto
FEUP 2010
Connect
Username
paulo
Full name
Paulo Andre Teixeira Pinto
Credit
240.5 €
Bets
bfail preferences
Store server IP
Server location is stored through multiple application executions
Save credentials
Credentials are automatically saved upon authentication
Auto login
Attempt to silently login when the application is launched
Auto synchronize
Synchronize information with the server every 30 seconds
5.0 €
Superliga - Fixture 3
2010-10-23 18:00
Braga 0 - 2 Benfica
1.0 €
Superliga - Fixture 3
2010-10-24 18:00
Sporting 1 - 2 Porto
3.4 €
Superliga - Fixture 4
2010-10-31 10:00
Sporting 1 - 2 Braga
10.0 €
Superliga - Fixture 4

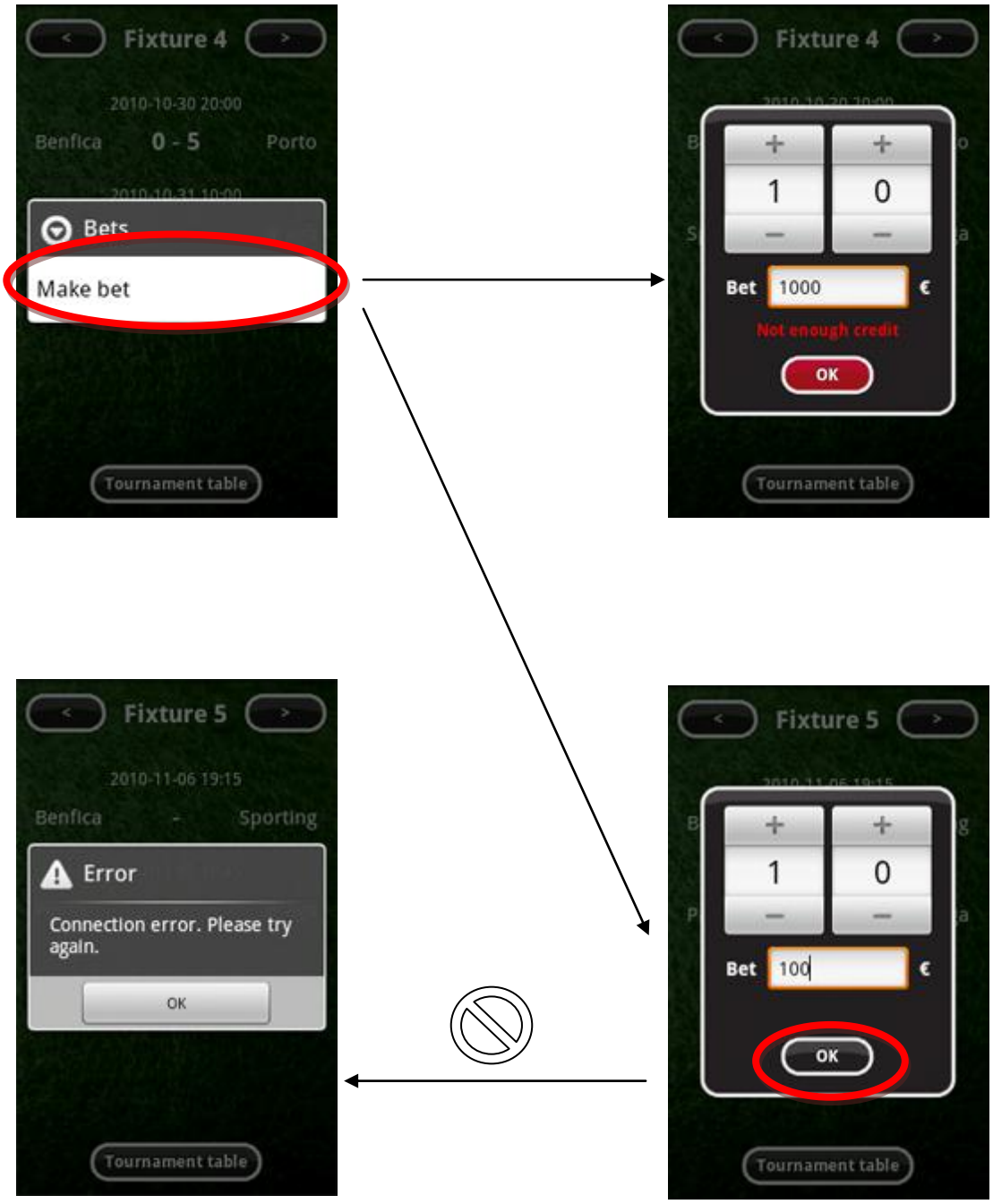

bfail
Superliga
Torneio de LGP
Campeonato das Aldeias
Disconnect
Fixture 3
2010-10-23 18:00
Braga 1 - 2 Benfica
2010-10-24 18:00
Sporting 3 - 1 Porto
Tournament table
Fixture 4
2010-10-30 20:00
Benfica - Porto
2010-10-31 10:00
Sporting - Braga
Tournament table
Fixture 4
2010-10-30 20:00
Benfica - Porto
Admin
Set match result
Resolve credit
Tournament table
Fixture 4
2010-10-30 20:00
Benfica 0 - 5 Porto
Bets
Make bet
Tournament table

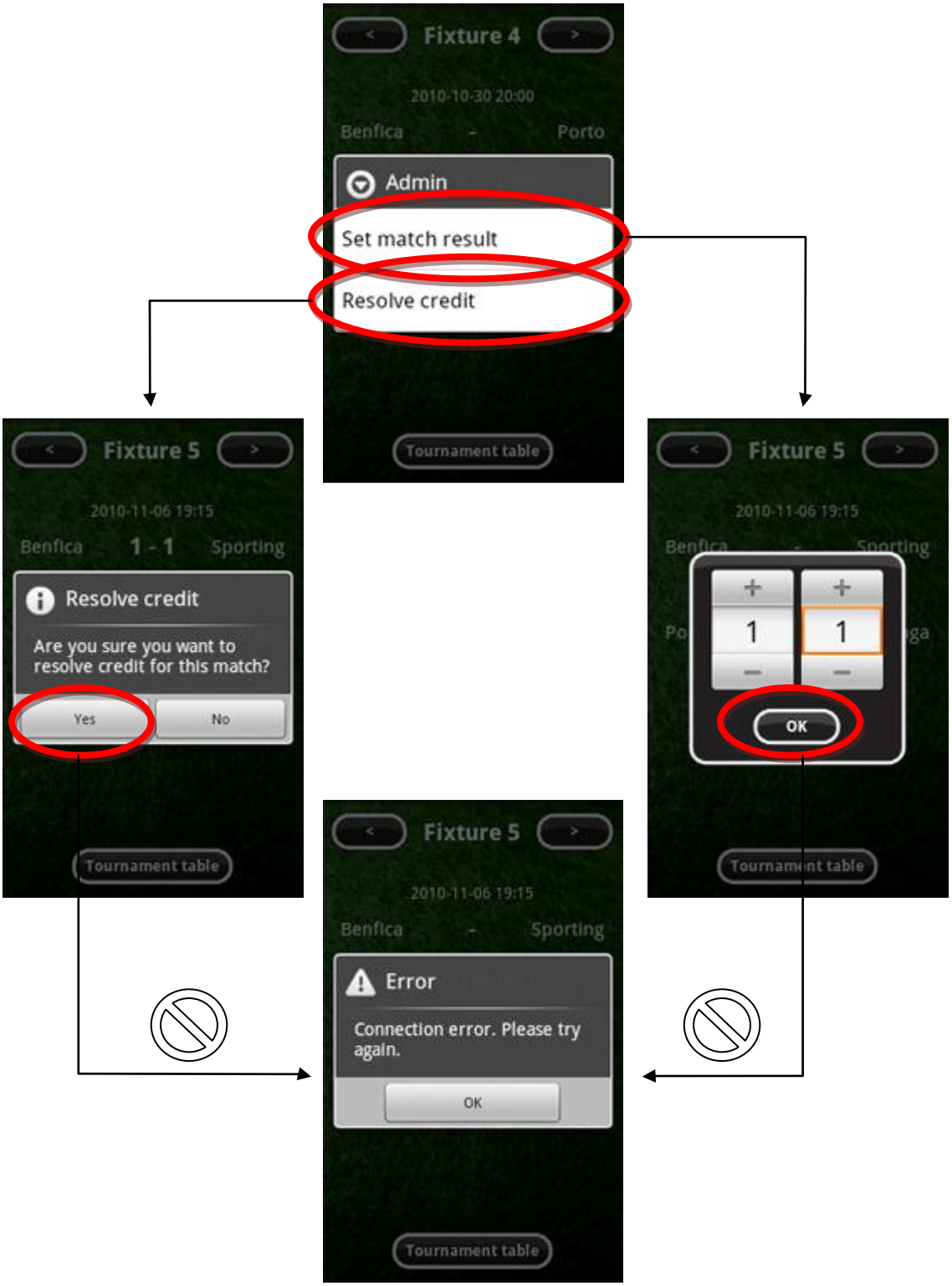

Team P M W D L GF GA
1 Porto 6 3 2 0 1 6 6
3 Braga 3 3 1 0 2 6 5
4 Benfica 3 3 1 0 2 3 4
All
Home
Away
Fixtures
Last
Next
Form
Position
Fixture 1
2010-10-08 19:00
Porto 2 - 1 Benfica
2010-10-16 18:00
Braga 2 - 3 Porto
Fixture 3
2010-10-24 18:00
Sporting 3 - 1 Porto
Fixture 4
2010-10-30 20:00
Benfica - Porto
Fixture 5
2010-11-07 19:45
Porto - Braga
Fixture 6
2010-11-14 20:30
Porto - Sporting
Fixture 4
2010-10-30 20:00
Benfica - Porto
Admin
Set match result
Resolve credit
Tournament table
Fixture 4
2010-10-30 20:00
Benfica 0 - 5 Porto
Bets
Make bet
Tournament table


Last
Next
Form
Position
Fixture 1
2010-10-08 19:00
Porto
2 - 1
Benfica
Fixture 2
2010-10-16 18:00
Braga
2 - 3
Porto
Fixture 3
2010-10-24 18:00
Sporting
3 - 1
Porto


Last
Next
Form
Position
Result
W
D
L
1
2
3
Fixture


Last
Next
Form
Position
Position
1
2
3
4
1
2
3
Fixture


Fixture 4
2010-10-30 20:00
Benfica
0 - 5
Porto
Bets
Make bet
Tournament table
Fixture 4
1
0
Bet
1000
€
Not enough credit
OK
Tournament table
Fixture 5
2010-11-06 19:15
Benfica
-
Sporting
Error
Connection error. Please try again.
OK
Tournament table
Fixture 5
1
0
Bet
100
€
OK
Tournament table


Fixture 4
2010-10-30 20:00
Benfica - Porto
Admin
Set match result
Resolve credit
Tournament table
Fixture 5
2010-11-06 19:15
Benfica 1 - 1 Sporting
Resolve credit
Are you sure you want to resolve credit for this match?
Yes
No
Tournament table
Fixture 5
2010-11-06 19:15
Benfica - Sporting
1
1
OK
Tournament table
Fixture 5
2010-11-06 19:15
Benfica - Sporting
Error
Connection error. Please try again.
OK
Tournament table

5. Conclusão

A plataforma *Android* ainda tem um mercado pouco extenso de aplicações, comparando, por exemplo, com a *Apple Store*. No entanto, a tecnologia tem vindo a evoluir desde o seu lançamento em 2005 e dado que a *framework* tem uma documentação extensa e uma boa comunidade, foi-nos bastante fácil construir uma aplicação que fosse intuitiva e visualmente agradável que tirasse partido de diversas funcionalidades que a plataforma disponibiliza.

O servidor de aplicações *GlassFish* a correr sobre *Apache Tomcat* permitiu manter uma implementação total do projecto em Java, tornando-o totalmente portável. Contudo, pôde-se verificar que a sua velocidade de execução não era a mais rápida, para além dos elevados gastos de memória. Possivelmente uma implementação mais *lightweight* baseada em *Django* e *Python* aumentaria o desempenho a nível do servidor.

Embora os objectivos do trabalho tenham sido totalmente alcançados, possíveis melhorias incluiriam uma interface para operações na base de dados (CRUD a nível do servidor), mais informação sobre as equipas, um número maior de estatísticas e gráficos e ainda a possibilidade de actualizar informações sobre um jogo a decorrer.

Bibliografia

[1] Burnette, Ed. Hello, Android, Third Edition. Pragmatic Bookshelf, 2010;

[2] Main - Jersey: RESTful Web services made easy. http://wikis.sun.com/display/Jersey/Main. Acedido em Outubro de 2010;

[3] Using your own SQLite Database in Android applications. http://www.reigndesign.com/blog/using-your-own-sqlite-database-in-android-applications/. Acedido em Outubro de 2010;

[4] Android Developers. http://developer.android.com/. Acedido em Novembro de 2010;

[5] Android Drawables. http://androiddrawableexplorer.appspot.com/. Acedido em Novembro de 2010;

[6] NumberPicker Widget for Android | Quietly Coding. http://www.quietlycoding.com/?p=5. Acedido em Novembro de 2010;

[7] GraphView - Android Application Development Tutorials and Free Components. http://android.arnodenhond.com/components/graphview. Acedido em Novembro de 2010.