Controle Multiusuário de Despesas Diárias para a Plataforma Android

Aluno(a): Bruno Muehlbauer de Souza

Orientador: Francisco Adell Péricas

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Especificação

Roteiro

- Implementação
- Operacionalidade da implementação
- Resultados e discussões
- Conclusão
- Extensões

Introdução

- Controle de despesas
- Gastos fantasmas
- Multiusuário

Objetivos

- Desenvolver um sistema de controle multiusuário de despesas diárias para a plataforma Android.

Objetivos Específicos

- Sincronizar informações entre um dispositivo móvel e base de dados remota por meio de um *webservice*.
- Lançar despesas e receitas através de uma interface Android.
- Permitir mais de um usuário controlar uma mesma conta compartilhada.

Fundamentação Teórica

- Planejamento financeiro
- Controle de gastos
- Webservice REST
- Android e Android Studio

Trabalhos Correlatos

- Minhas Economias
- Mobills

Minhas Economias

- Web e mobile
- Android, IOS e Windows Phone
- Verifica saldos
- Despesas e receitas
- Transferências
- Parcelamento
- Anotações

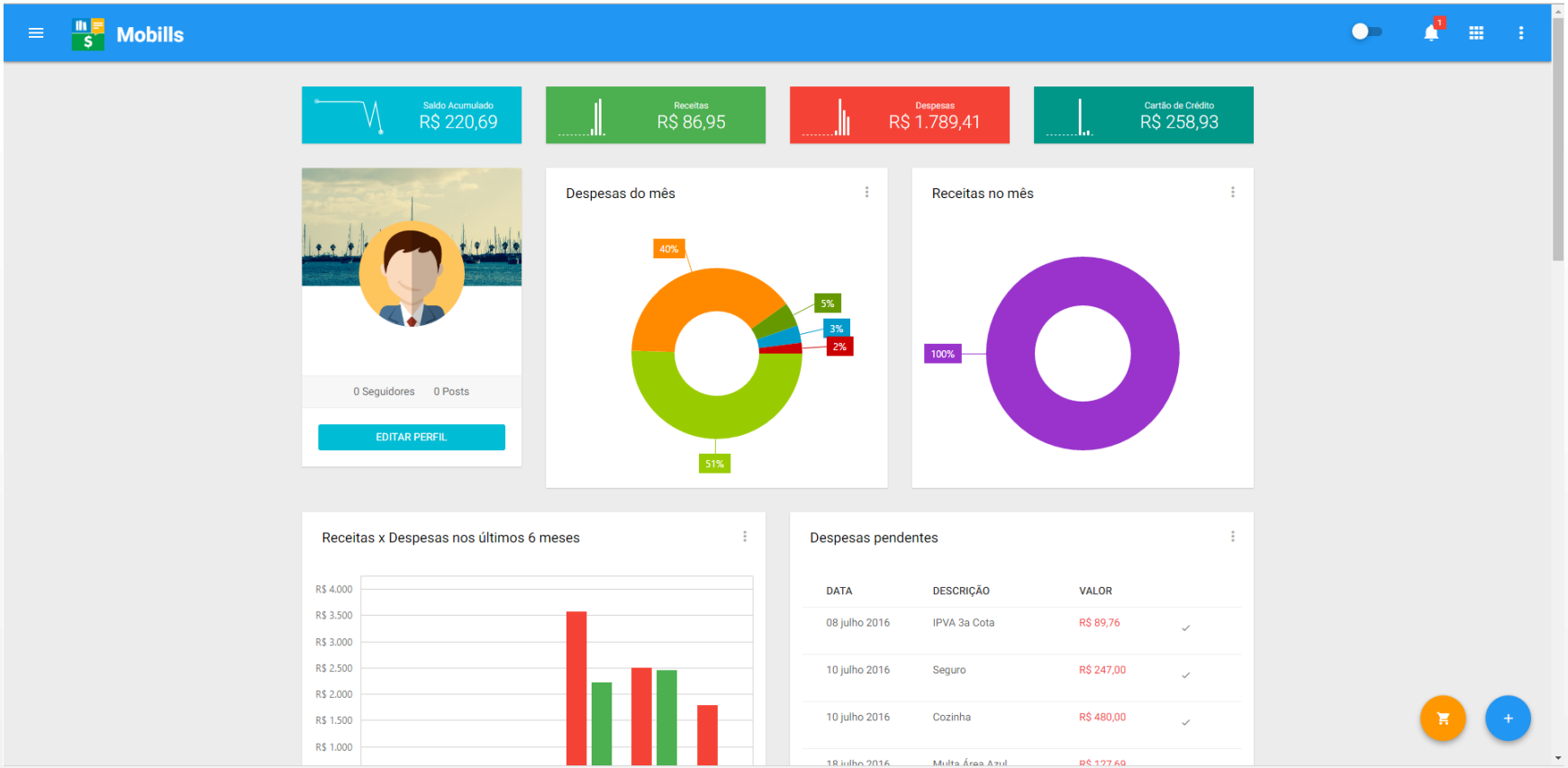
Minhas Economias

FURB
UNIVERSIDADE DE BLUMENAU

Mobills

- Web e mobile
- Android, IOS e Windows Phone
- Contas e Transferências
- Análise inteligente de despesas
- Metas e orçamentos
- Calendário
- Estatísticas
- Exportação para Excel e importação via SMS.

Mobills



FURB
UNIVERSIDADE DE BLUMENAU

Mobills


17:18
Principal
14
Visão geral - julho
Contas
Saldo acumulado
R$ 220,69
Receitas
R$ 86,95
Despesas
R$ 1.789,41
Cartão de Crédito
R$ 258,93
Pendências e Alertas
7
Despesas pendentes
R$ 1.199,54
1
Receitas pendentes
R$ 86,95
3
SMS para importar
+

Especificação

- Requisitos funcionais
- Requisitos não-funcionais
- Casos de uso
- Arquitetura
- Diagramas de classe

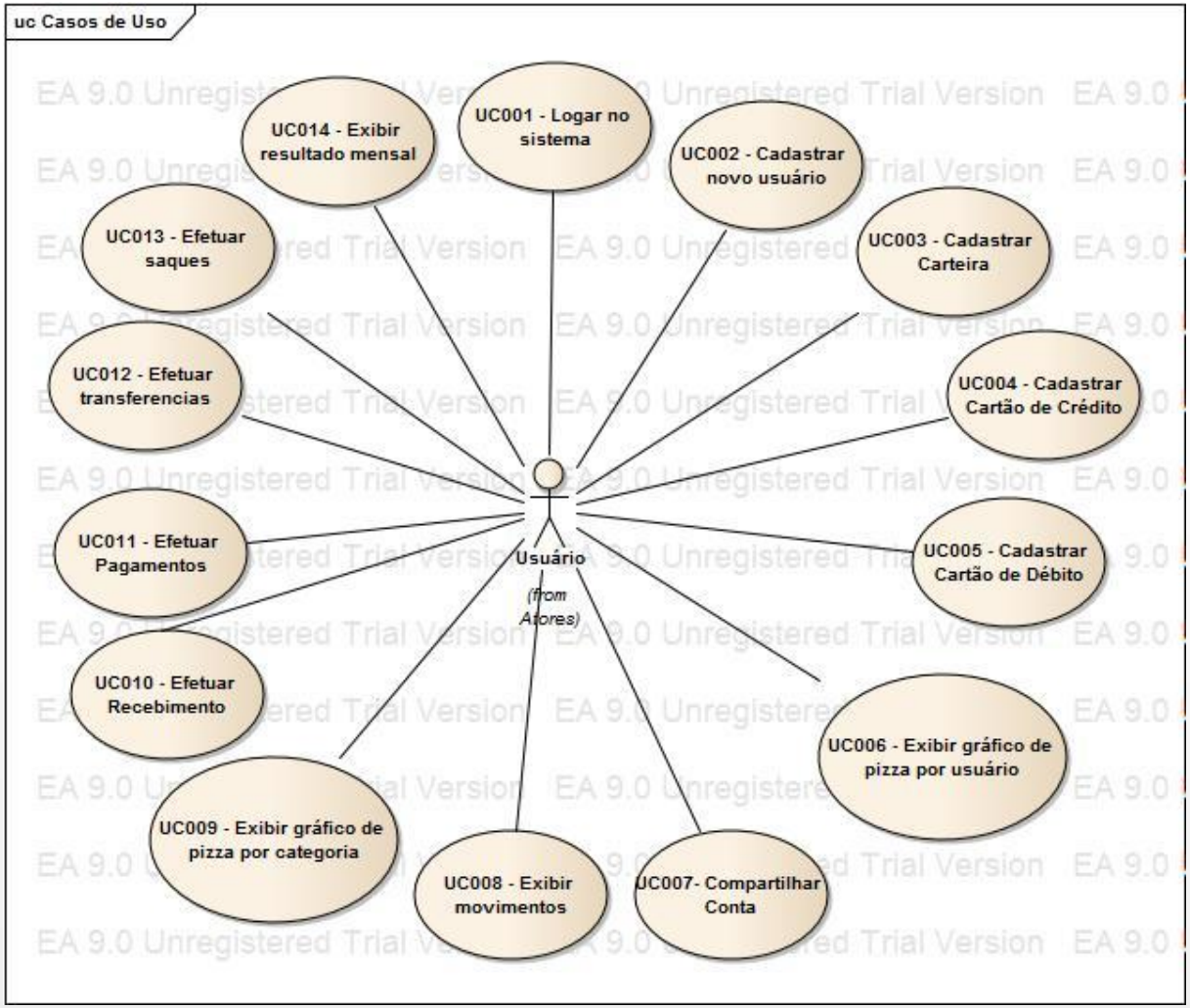
Requisitos Funcionais

- RF010 – O sistema móvel permitirá movimentar contas por meio de lançamentos de Receita ou Despesa.
- RF015 – O sistema móvel permitira o compartilhamento de contas com outros usuários.
- RF017 – O sistema móvel permitirá visualizar extratos de conta.

Requisitos Não-Funcionais

- RNF001 – O sistema móvel deverá rodar na plataforma Android.
- RNF002 – O sistema móvel deverá enviar informações por meio de webservices.
- RNF003 – A versão mínima do Android para o sistema móvel é a 4.4.2.

Casos de Uso



Arquitetura

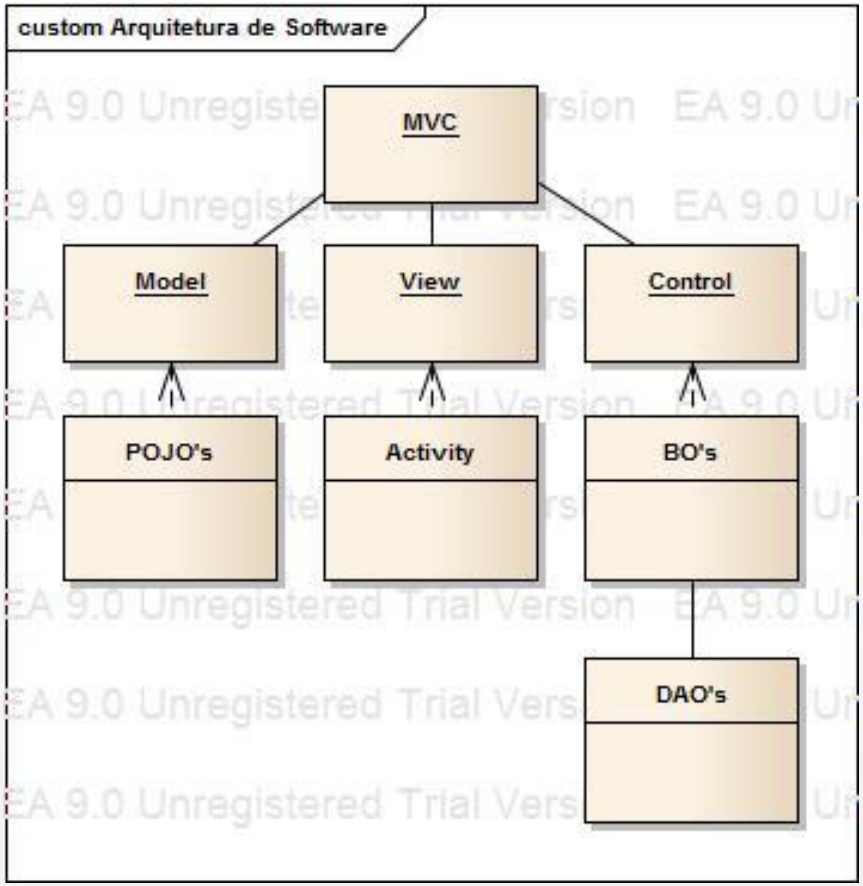


Diagrama de Classes - Webservice

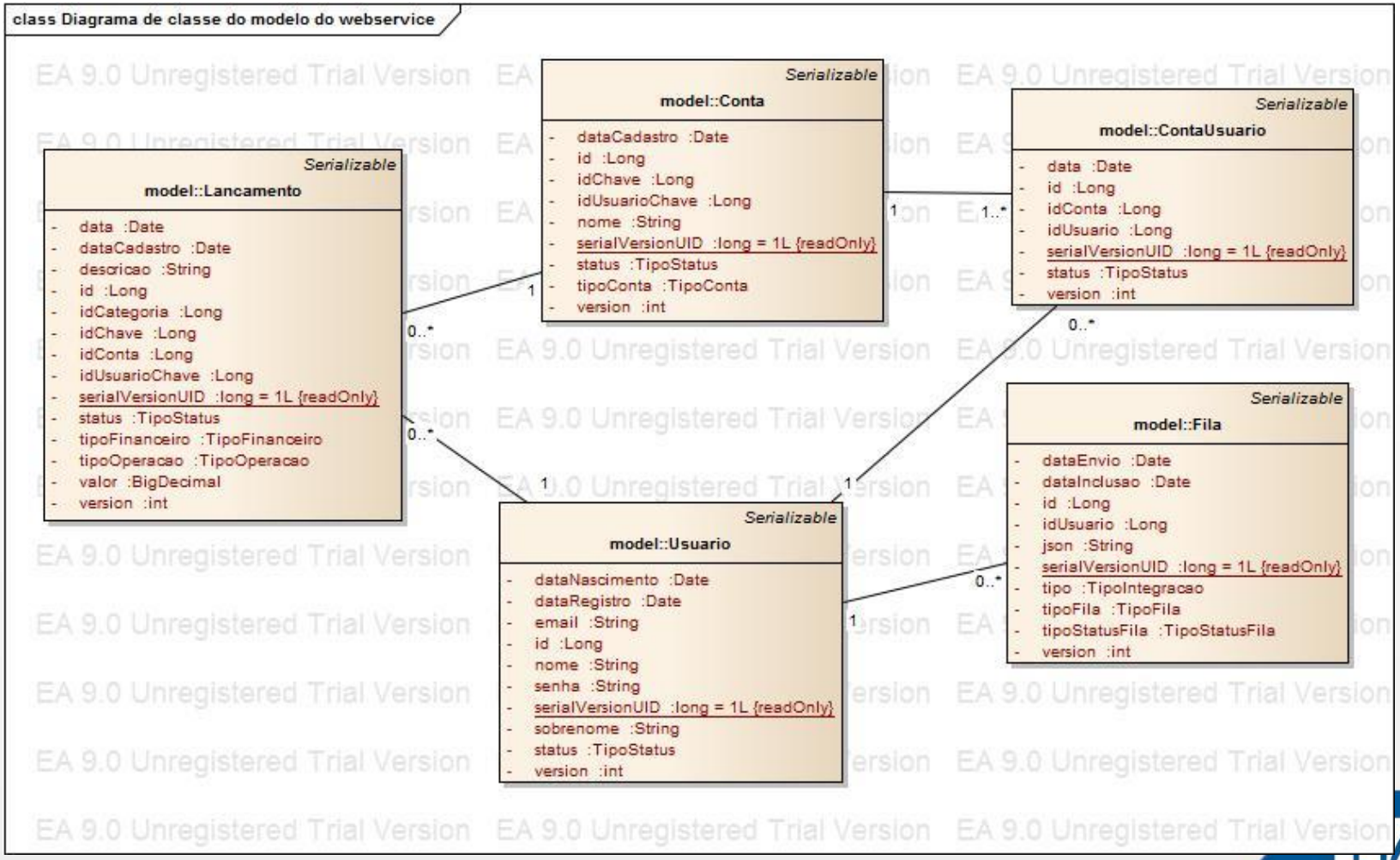
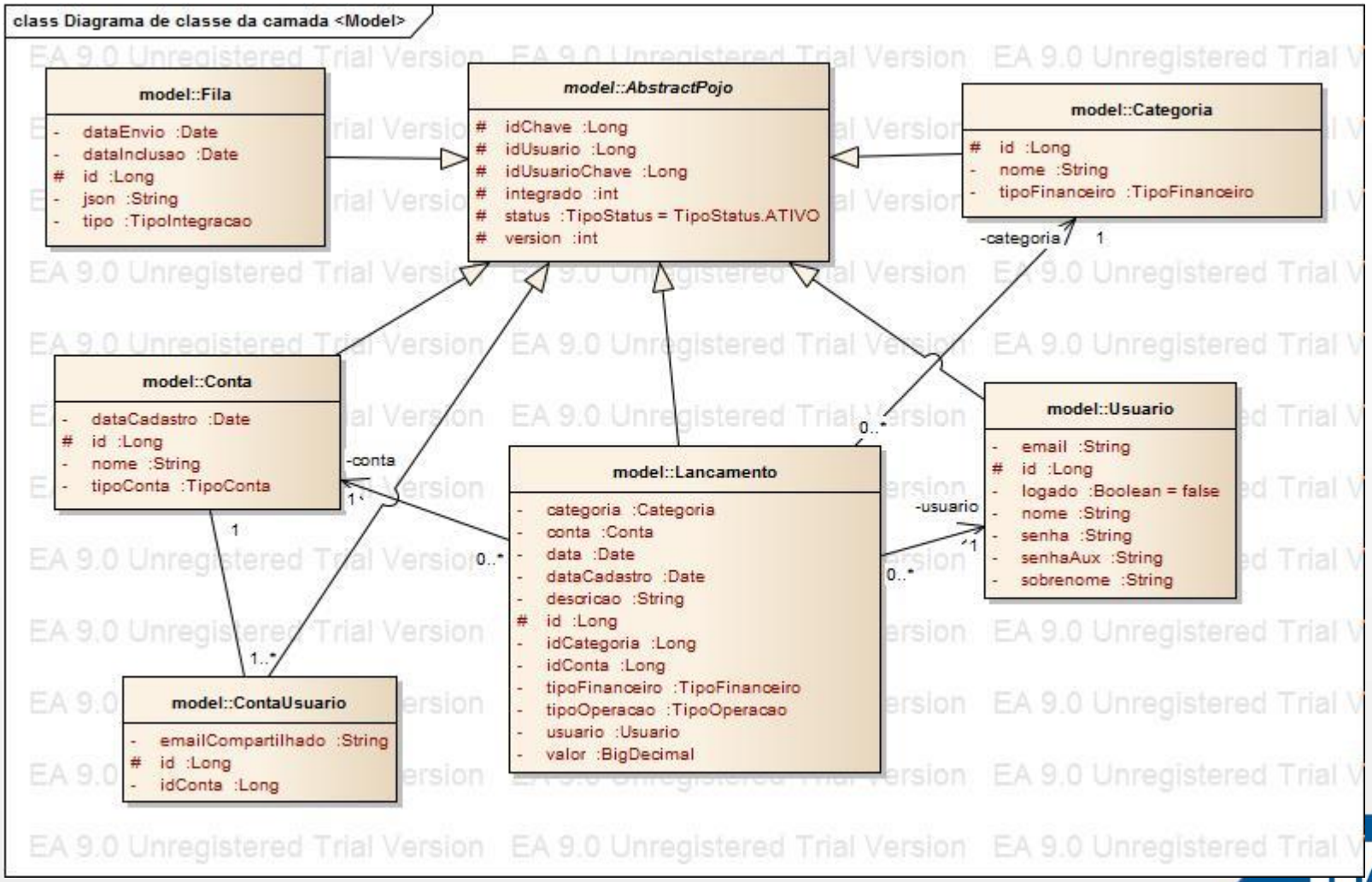
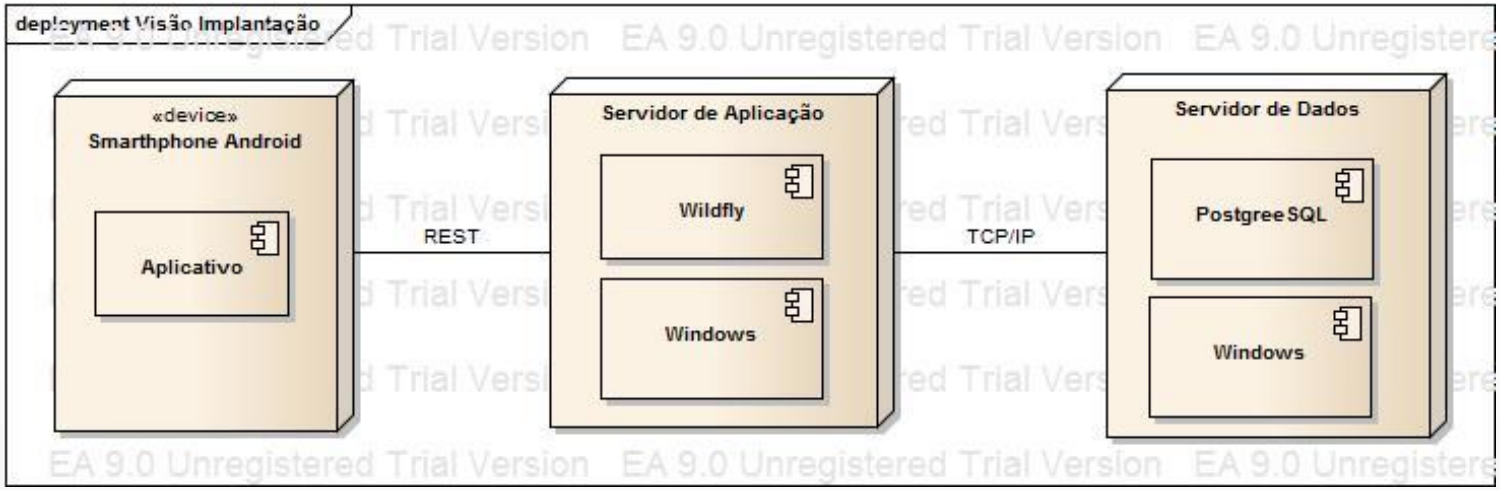


Diagrama de classes - Mobile



Implementação



Persistência - Mobile

Quadro 1 - Classe `Lancamento` do Software *mobile*

```
@DatabaseTable(tableName = "Lancamento")
public class Lancamento extends AbstractPojo {

   @Expose
   @DatabaseField(generatedId = true)
   protected Long id;

   @DatabaseField(width = 40)
   private String descricao;

   @DatabaseField
   private TipoFinanceiro tipoFinanceiro;

   @DatabaseField
   private Date data;

   @DatabaseField
   private Date dataCadastro;

   @DatabaseField
   private BigDecimal valor;

   @DatabaseField
   private Long idCategoria;

   @DatabaseField
   private Long idConta;
}
```

Persistência - Webservice

Quadro 2 - Classe `Conta` do webservice

```
@Entity
@XmlRootElement
public class Conta implements Serializable {

        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name = "id", updatable = false, nullable = false)
        private Long id;

        @Version
        @Column(name = "version")
        private int version;

        @Column
        private String nome;

        @Convert(converter = TipoContaEnumConverter.class)
        private TipoConta tipoConta;

        @Column
        @Temporal(TemporalType.TIMESTAMP)
        private Date dataCadastro;

        @Column
        private Long idChave;

        @Column
        private Long idUsuarioChave;

        @Convert(converter = TipoStatusEnumConverter.class)
        private TipoStatus status;
}
```

Persistência - Mobile

Quadro 4 - Classe `DataBaseHelper` responsável pela criação e atualização da base.

```
public class DatabaseHelper extends OrmLiteSqliteOpenHelper {

   private static final String DATABASE_NAME = "coddi.sqlite";
   private static final int DATABASE_VERSION = 137;

   public DatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }

   @Override
   public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {
      try {
         TableUtils.createTable(connectionSource, Usuario.class);
         TableUtils.createTable(connectionSource, Categoria.class);
         TableUtils.createTable(connectionSource, Conta.class);
         TableUtils.createTable(connectionSource, Lancamento.class);
         TableUtils.createTable(connectionSource, Fila.class);
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }

   @Override
   public void onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource, int oldVersion, int
newVersion) {
      try {
         TableUtils.dropTable(connectionSource, Usuario.class, true);
         TableUtils.dropTable(connectionSource, Categoria.class, true);
         TableUtils.dropTable(connectionSource, Conta.class, true);
         TableUtils.dropTable(connectionSource, Lancamento.class, true);
         TableUtils.dropTable(connectionSource, Fila.class, true);

         onCreate(database, connectionSource);
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Persistência - Webservice

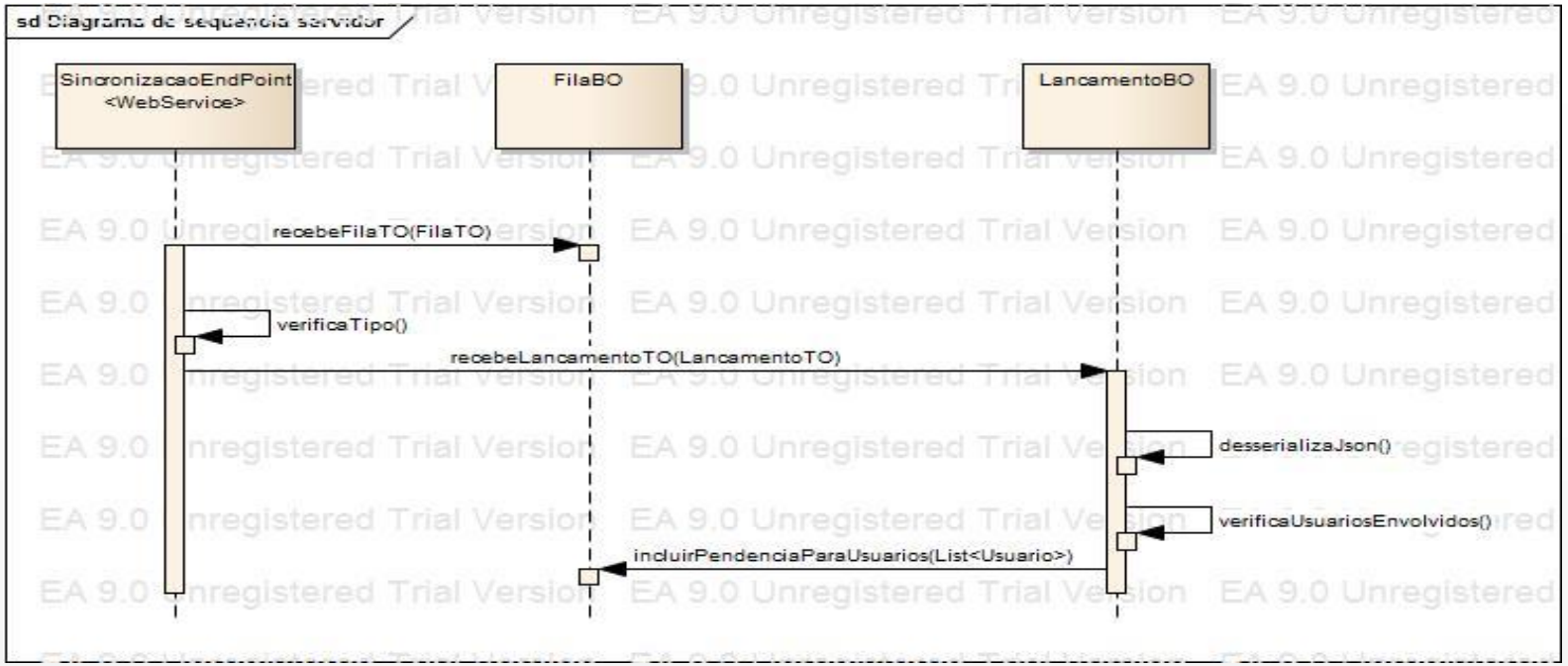
Quadro 5 - Persistence.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.1"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="Coddi-persistence-unit" transaction-type="JTA">
    <description>Forge Persistence Unit</description>
    <jta-data-source>java:jboss/datasources/postgresql</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action"
            value="none" />
      <property name="javax.persistence.schema-generation.scripts.action" value="drop-and-create"/>
      <property name="javax.persistence.schema-generation.scripts.create-target" value="CoddiCreate.ddl"/>
      <property name="javax.persistence.schema-generation.scripts.drop-target" value="CoddiDrop.ddl"/>
    </properties>
  </persistence-unit>
</persistence>
```

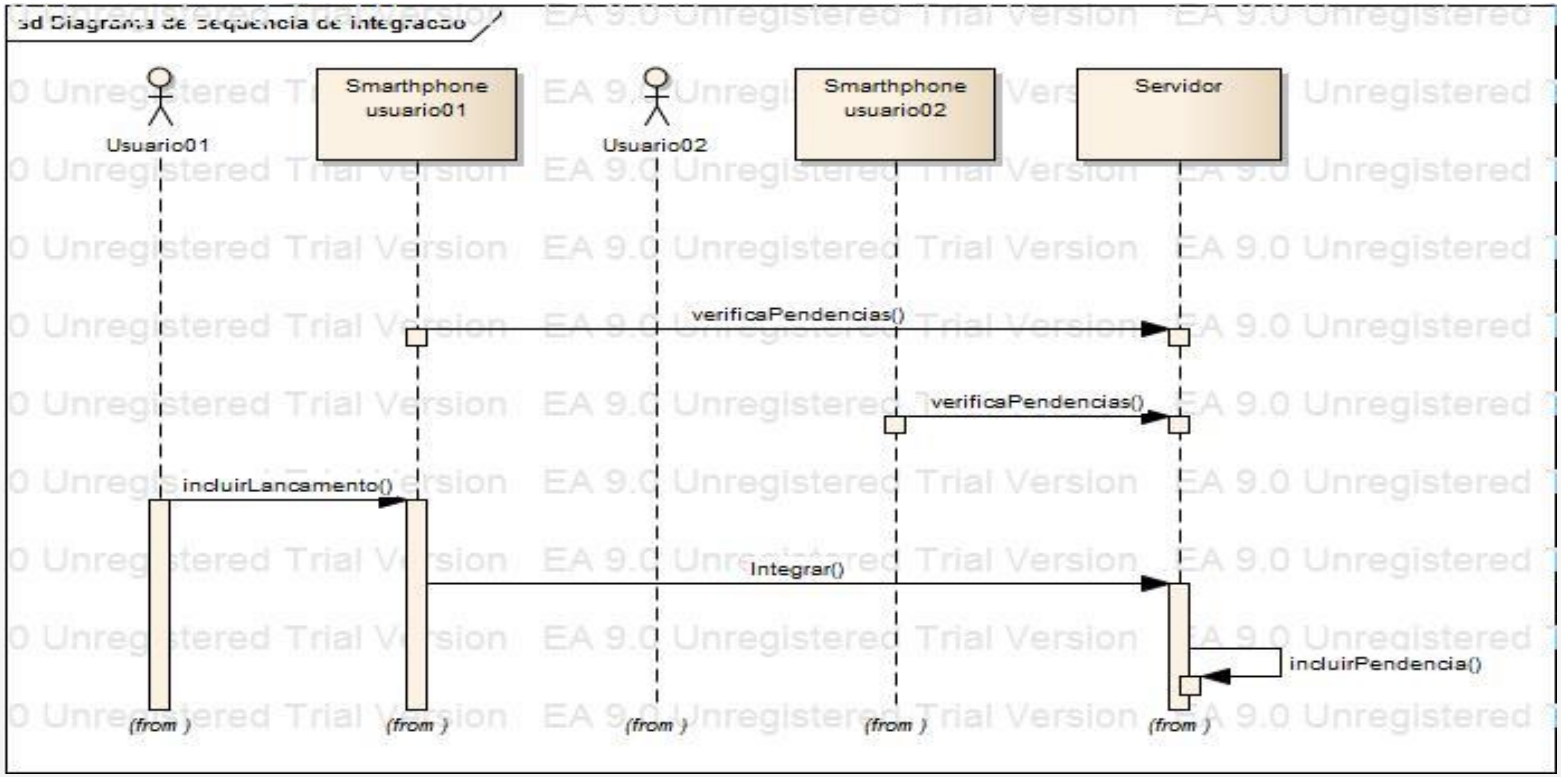
Integração

- Funcionalidade
  - Colunas chave
  - Código de referência
- Prós
- Contras

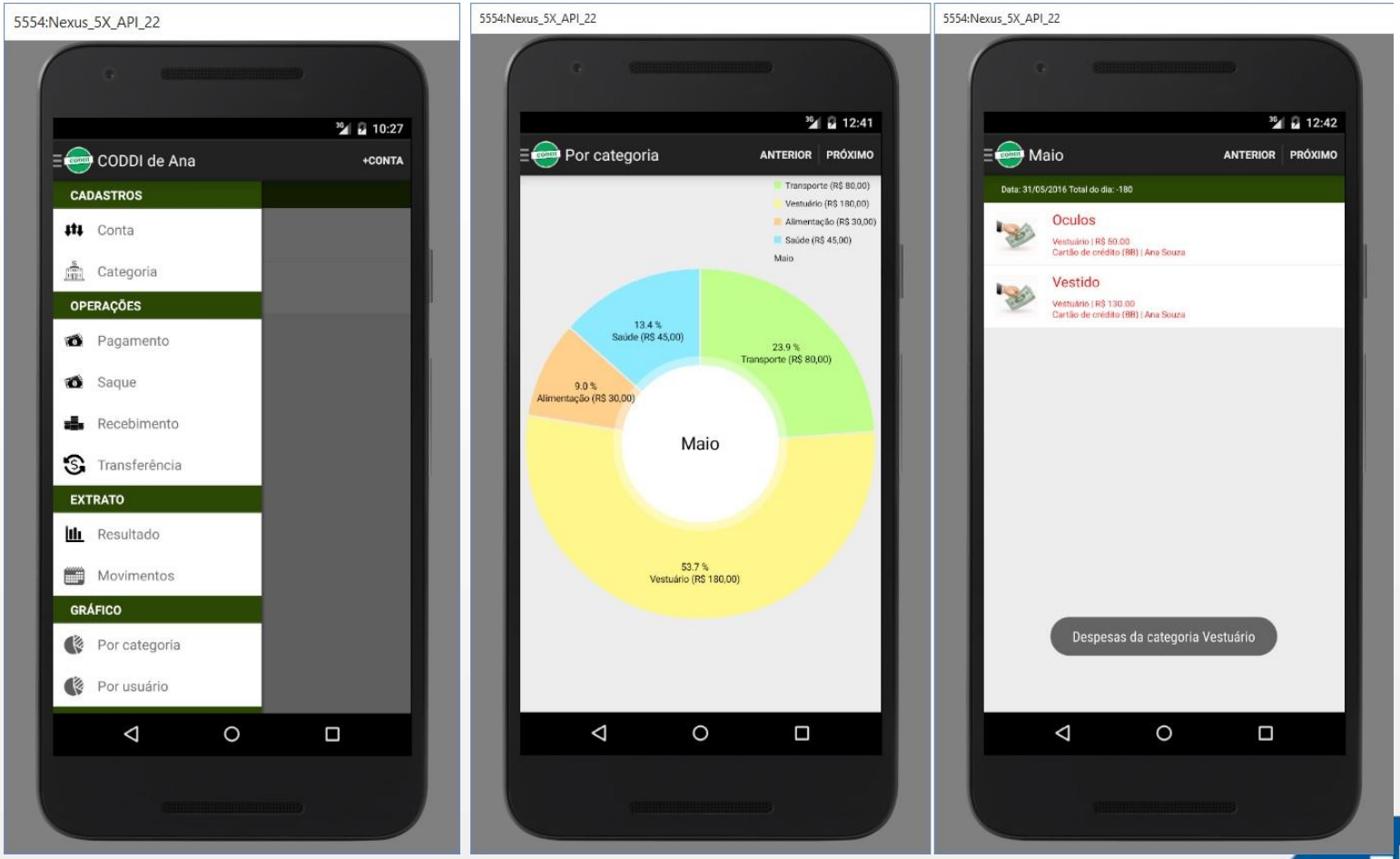
Processo de integração



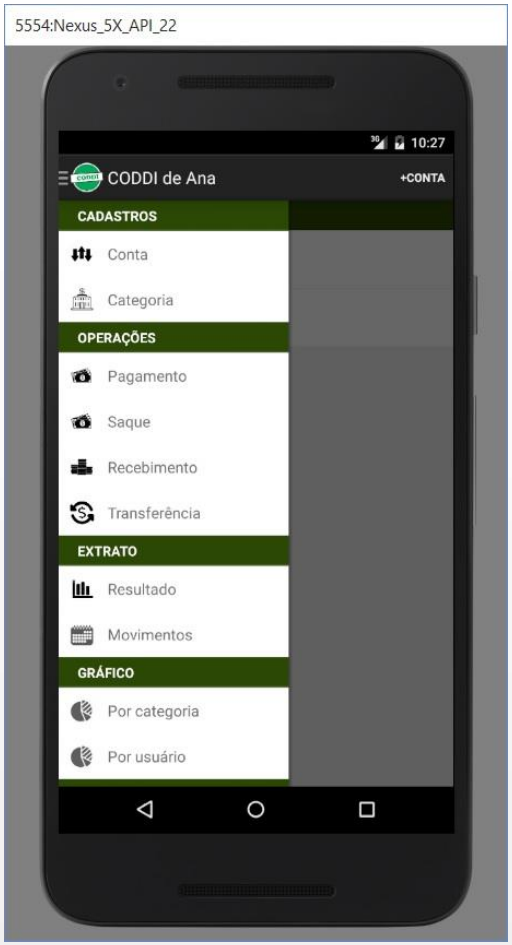
Integração multiusuário

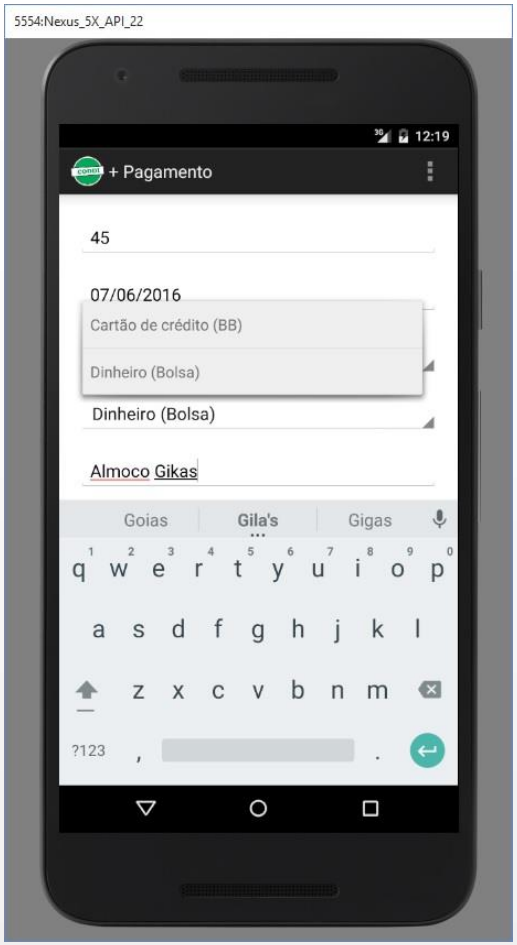


Operacionalidade da Implementação

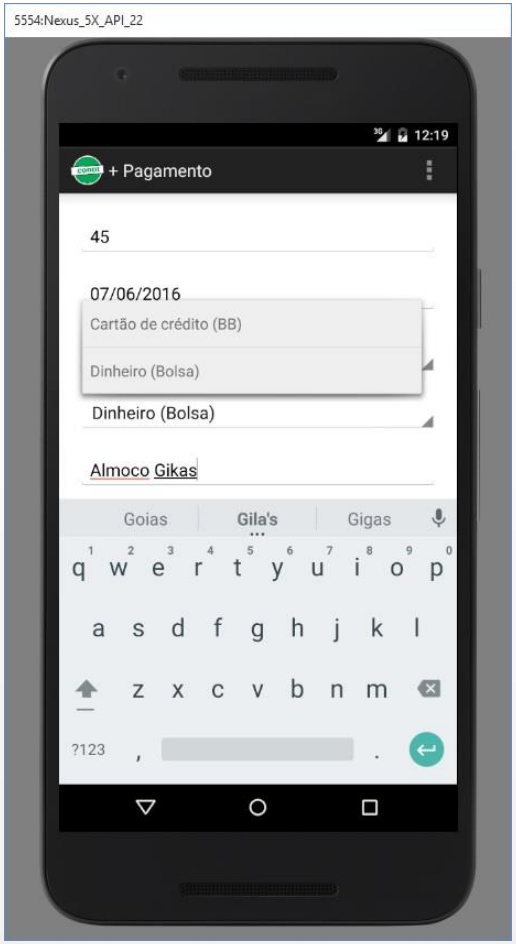


Operações


5554:Nexus_5X_API_22
10:27
CODDI de Ana
+CONTA
CADASTROS
Conta
Categoria
OPERAÇÕES
Pagamento
Saque
Recebimento
Transferência
EXTRATO
Resultado
Movimentos
GRÁFICO
Por categoria
Por usuário


5554:Nexus_5X_API_22
12:19
+ Pagamento
45
07/06/2016
Cartão de crédito (BB)
Dinheiro (Bolsa)
Dinheiro (Bolsa)
Almoco Gikas
Goias
Gila's
Gigas

Formas de pagamento



Resultados e Discussões

- Sucesso no atendimento dos requisitos e do objetivo proposto
- Contemplou a sincronização dos dados, replicação e inclusão em outro dispositivo
- Mostrou-se funcional em ambiente de testes, no entanto, precisaria de ajustes para funcionar via Google Play

Conclusões e Sugestões

- Processos e validações com alta complexidade
- Não é possível contemplar todas as situações que um aplicativo comercial contempla
- Desafio de integrar sem código de referência

Extensões

- Integração das compras feitas com cartão via SMS
- Importar valor através da câmera do smarthphone de comprovantes fiscais e não fiscais
- Adaptação do aplicativo para Google Play

OBRIGADO!