

**DIEGO D'LEON NUNES  
DIÓGENES APARECIDO REZENDE**

**APLICATIVO PARA CONSULTA DE NOTAS**

**UNIVERSIDADE DO VALE DO SAPUCAÍ  
POUSO ALEGRE – MG**

**2015**

**SUMÁRIO**

**1**            **QUADRO METODOLÓGICO ..... 2**

**1.1**        **Tipo de pesquisa ..... 2**

**1.2**        **Contexto de pesquisa ..... 2**

**1.3**        **Instrumentos ..... 3**

**1.4**        **Procedimentos ..... 4**

**REFERÊNCIAS..... 13**

# 1 QUADRO METODOLÓGICO

Nesse capítulo serão apresentados os métodos adotados para se realizar a pesquisa, tais como tipo de pesquisa, contexto, procedimentos, entre outros.

## 1.1 Tipo de pesquisa

Uma pesquisa é o ato de buscar e procurar pela resposta de algo. Marconi e Lakatos (2002, p. 15) definem pesquisa como “uma indagação minuciosa ou exame crítico e exaustivo na procura de fatos e princípios”.

Existem diversos tipos de pesquisa, no entanto percebeu-se que para o propósito desta, a mais indicada foi a pesquisa aplicada, pois está se desenvolvendo um projeto real que poderá ser utilizado por qualquer instituição de ensino, mas que não mudará a forma com que as pessoas recebam suas informações, apenas acrescenta mais uma forma de consultá-las.

Segundo Marconi e Lakatos (2002, p. 15), uma pesquisa do tipo aplicada “caracteriza-se por seu interesse prático, isto é, que os resultados sejam aplicados ou utilizados, imediatamente, na solução de problemas que ocorrem na realidade”.

Dessa maneira, percebeu-se que a pesquisa enquadra-se no tipo de pesquisa aplicada, pois com a execução da mesma resolve-se um problema específico, e para isso está desenvolvendo-se um aplicativo para dispositivos móveis que facilitará aos graduandos acessarem o sistema *web* de uma universidade.

## 1.2 Contexto de pesquisa

Essa pesquisa será benéfica a qualquer instituição educacional que possua um portal *online*, pois facilitará o acesso dos discentes às suas informações escolares.


Está em fase de criação um aplicativo para dispositivos móveis, porém inicialmente apenas para a plataforma *Android*, o qual notifica os usuários quando há alguma mudança, como por exemplo, ao ser lançada uma nota.

O aluno consegue acessar o aplicativo com o mesmo *login* do sistema *web*. O utilitário acessa o *webservice* que é responsável por buscar as informações no banco de dados e apresentá-las no dispositivo.

### **1.3 Instrumentos**

Pode-se dizer que um questionário é uma forma de coletar informações através de algumas perguntas feitas a um público específico. Segundo Gunther (2003), questionário pode ser definido como um conjunto de perguntas que mede a opinião e interesse do respondente.

Foi realizado um questionário simples, que está apresentado na figura 1, contendo quatro perguntas e enviado para *e-mails* de alguns alunos da universidade. O foco desse questionário era saber o motivo pelo qual os usuários mais acessavam o portal do aluno e se tinham alguma dificuldade em encontrar o que procuravam. Obteve-se um total de treze respostas, no qual pode-se perceber que a maioria dos entrevistados afirmam terem dificuldades para encontrar o que precisam e que o sistema não avisa quando ocorre alguma alteração. Sobre o motivo do acesso cem por cento respondeu que entram no sistema web para consultar suas notas.



### Pesquisa sobre o portal do aluno

Qual é sua opinião sobre o portal do aluno?

☐ Ótimo  
☐ Bom  
☐ Ruim  
☐ Péssimo

Qual é sua maior dificuldade para acessar o portal do aluno?

☐ Não tenho acesso a internet  
☐ Demoro para encontrar o que preciso  
☐ O sistema não avisa quando são lançadas as notas  
☐ Outro:

A maior parte das vezes que acesso o portal do aluno é para?

☐ Ver minhas notas  
☐ Ver provas agendadas  
☐ Ver minhas faltas  
☐ Buscar contatos dos professores  
☐ Consultar financeiro  
☐ Consultar material postado pelos professores  
☐ Outro:

Você acha que um aplicativo para celular para acessar o portal seria?

☐ Ótimo  
☐ Bom  
☐ Ruim  
☐ Péssimo

100% concluído

**Figura 1** – Questionário Aplicado. **Fonte:**Elaborado pelos autores.

Outra forma utilizada para realizar a pesquisa foram as reuniões, ou seja, unir-se com uma ou mais pessoas em um local, físico ou remotamente para tratar algum assunto específico. Para Ferreira (1999), reunião é o ato de encontro entre algumas pessoas em um determinado local, com finalidade de tratar qualquer assunto.

Durante a pesquisa, foram realizadas reuniões entre os participantes com o objetivo de discutir o andamento das tarefas pela qual cada integrante ficou responsável. Além disso entravam em discussão, nessas reuniões, o cumprimento das metas propostas por cada participante e o estabelecimento de novas metas. Foi utilizada nessa pesquisa, referências de livros, revistas, manuais e web sites.

## 1.4 Procedimentos

O primeiro procedimento realizado para chegar a pesquisa proposta, foi planejar o *software* através da linguagem UML, que necessitou a instalação da ferramenta *Astah* na sua versão 6.8.0.37.

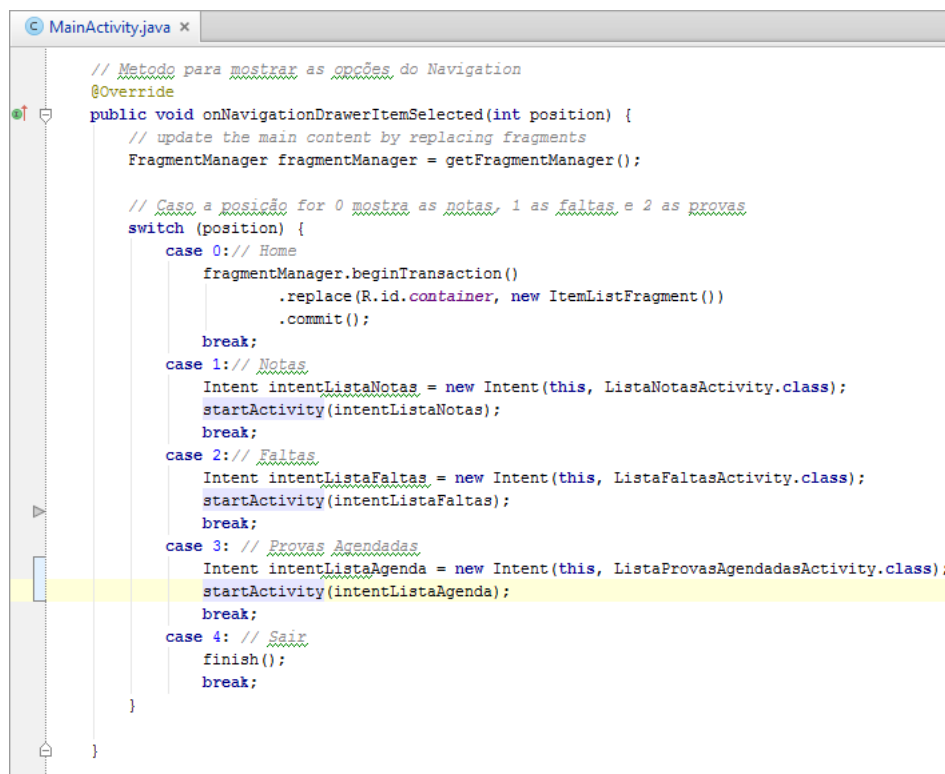
Como o planejamento é um passo importante, para o aplicativo Android, fez-se necessários os diagramas de classe, de caso de uso principal e de atividade.

Para a construção do *software* proposto por essa pesquisa além do levantamento de requisitos que é peculiar da construção de qualquer *software*,

- Diagrama de casos de uso;
- Diagrama de sequência;
- Diagrama de Entidade e Relacionamento (ou Modelo de Entidade e Relacionamento )

Para iniciar o desenvolvimento do aplicativo, primeiramente fez-se necessária a instalação e configuração da plataforma *Android Studio* versão 1.1.0 e *Android SDK* versão 24.0.2. Ao concluir essa tarefa, deu-se o início ao aplicativo *Android*.

O primeiro passo foi a criação de uma *activity main*, denominada *MainActivity*, a qual será executada quando a aplicação for iniciada. O tipo de *activity* escolhida é chama-se *Navigation Drawer Layout*. Na figura 2 pode-se ver o método `onNavigationDrawerItemSelected()`, que tem por finalidade mostrar as opções de navegação e o que deverá acontecer quando uma delas for clicada.



```
// MainActivity.java
// Metodo para mostrar as opções do Navigation
@Override
public void onNavigationDrawerItemSelected(int position) {
    // update the main content by replacing fragments
    FragmentManager fragmentManager = getFragmentManager();

    // Caso a posição for 0 mostra as notas, 1 as faltas e 2 as provas
    switch (position) {
        case 0: // Home
            fragmentManager.beginTransaction()
                .replace(R.id.container, new ItemListFragment())
                .commit();
            break;
        case 1: // Notas
            Intent intentListaNotas = new Intent(this, ListaNotasActivity.class);
            startActivity(intentListaNotas);
            break;
        case 2: // Faltas
            Intent intentListaFaltas = new Intent(this, ListaFaltasActivity.class);
            startActivity(intentListaFaltas);
            break;
        case 3: // Provas Agendadas
            Intent intentListaAgenda = new Intent(this, ListaProvasAgendadasActivity.class);
            startActivity(intentListaAgenda);
            break;
        case 4: // Sair
            finish();
            break;
    }
}
```

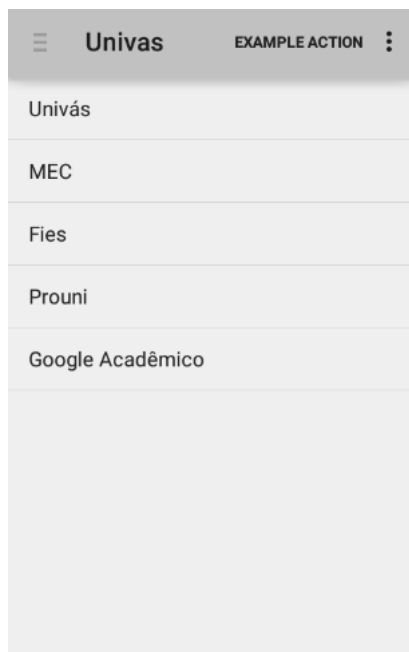
**Figura 2** – Implementação do método `onNavigationDrawerItemSelected()`. **Fonte:**Elaborado pelos autores.

Ao executa-la, funcionará como um menu, com as opções *Home*, *Notas*, *Faltas*, *Provas Agendadas* e *Sair*. Essas opções ficam escondidas e só apareceram quando clicado no canto superior esquerdo como mostra a figura 3.



**Figura 3** – Representação Visual de `onNavigationItemSelectedListener()`. **Fonte:**Elaborado pelos autores.

Sempre que a aplicação é iniciada ou quando o usuário retorna para a *Home*, é apresentada uma lista com acesso a sites úteis. Para que essa lista aparecesse foi utilizada uma *activity* do tipo *Master/Detail Flow* que já traz consigo o *widget* de *ListView*. Essa *activity* mostrará as seguintes opções: Univás, MEC, Fies, Prouni e Google Acadêmico. Ao clicar em um desses itens, será chamada uma atividade que mostrará os detalhes do item escolhido, nesse caso na *activity* que detalhará o item foi utilizada um *widget WebView* que mostrará a página *web* no espaço reservado. Dessa maneira, quando for clicado na opção Univás será carregada o site da universidade no aplicativo. Na figura 4, é possível ver a tela *Home* do aplicativo.



**Figura 4** – *Activity Home* do aplicativo. **Fonte:**Elaborado pelos autores.

Logo após foram criadas três novas *activities* do tipo *Blank Activity*, as quais listarão as matérias cursadas pelo aluno. Nelas foram inseridas o widget *ExpandableListView*, para que quando clicado em um item listado, ele expandido mostre seus detalhes. Com isso quando se escolhe o menu notas, será apresentada uma *activity* que listará todas as disciplinas cursadas e ao clicar em uma delas mostrará as notas referente a ela. Na figura 5, é possível ver uma *activity*, que lista algumas matérias e ao clicar em uma delas foi possível ver as notas.



**Figura 5** – *Activity Notas*. **Fonte:**Elaborado pelos autores.



Foi criada uma classe chamada *BuscaDados*, que tem por finalidade receber os dados do *webservice*, salvá-los no banco de dados local do aplicativo e os entregar para as classes que implementam o *Adapter* para listar as informações na tela do dispositivo.

No arquivo *AndroidManifest.xml* foi necessário alterar a opção *Android:icon*, que define qual será o ícone do aplicativo, por padrão ele apresenta o mascote do *Android*, no entanto foi definida uma imagem do logo da universidade. Foi necessário também incluir uma tag de *uses-permission*, que obriga o usuário a permitir o uso da *internet* pelo aplicativo, conforme pode ser visto na figura 6. Pode-se perceber que cada *activity* encontra-se dentro das tags *<activity><activity>* e que a *activity main*, deve conter a tag *<intent-filter>* e dentro dela *<action android:name="android.intent.action.MAIN"/>*, indicando que ela será a primeira a executar e *<category android:name="android.intent.category.LAUNCHER"/>*, informando que ficará disponível junto aos outros aplicativos do dispositivo.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diego.univas" >

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/univas"
        android:label="Univas"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="Univas" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

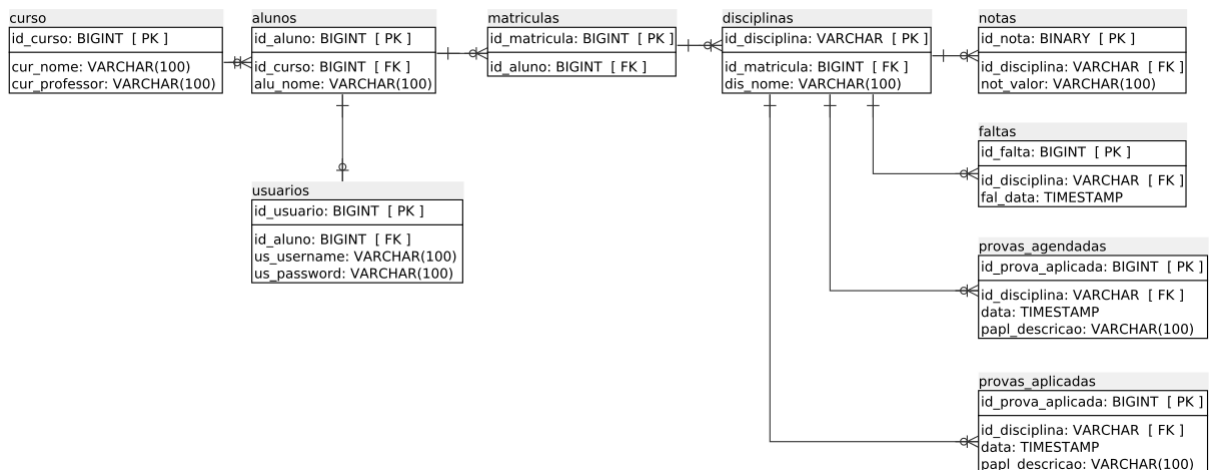
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".SettingsActivity"
            android:label="SettingsActivity"
            android:parentActivityName=".MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.diego.univas.MainActivity" />
        </activity>
        <activity
            android:name=".ListaNotasActivity"
            android:label="Notas"
            android:parentActivityName=".MainActivity" >
            <meta-data
```

**Figura 6** – *AndroidManifest.xml*. Fonte:Elaborado pelos autores.

O *Android Studio* tem uma facilidade de se trabalhar com controladores de versão, nesse caso foi escolhido o *GitHub*. Nele foi criado uma pasta e compartilhada entre os participantes e por fim configurou-se o *Git* com a IDE para que cada um possa ter a versão mais atualizada do projeto.

No que diz respeito à construção do *webservice*, foi necessário a instalação e configuração de um ambiente de desenvolvimento compatível com as necessidades apresentadas pelo *software* e que foram levantadas através dos requisitos. Foi instalado o *Servlet Container Apache Tomcat* em sua versão de número 7. O *Servlet Container* foi instalado para que o *Web Service* pudesse fornecer os serviços necessários para o consumo de dados do Aplicativo *Android*, haja vista que *Apache tomcat* faz uso amplo do protocolo HTTP<sup>1</sup> e da plataforma *Java* de desenvolvimento.

Para armazenar os dados gerados e/ou recebidos foi necessário fazer a instalação do Sistema Gerenciador de Banco de Dados(SGBD) *PostGreSql* na sua versão de número 9.2. Através de um levantamento de requisitos parciais e das reuniões entre os participantes foi possível construir um Diagrama de Entidade e Relacionamento, no qual ficou definido a estrutura do banco de dados da aplicação. A figura 7 mostra o Diagrama de Entidade e Relacionamento concebido para esta pesquisa.



**Figura 7** – Diagrama de Entidade e Relacionamento. **Fonte:**Elaborado pelos autores.

Fazendo uso desse diagrama foi possível criar todas as classes *Java* que representam as entidades do mapeamento objeto-relacional. Essas classes foram criadas fazendo uso de anotações próprias do *Hibernate*, que é um *framework* que implementa a especificação JPA<sup>2</sup>. Essas classes fazem parte dos mecanismos de persistência de dados, e são simplesmente POJO's<sup>3</sup> ou seja objetos simples que contêm somente atributos privados e os métodos *getters* e *setters* que servem apenas para encapsular estes atributos. Uma das classes criadas, foi a classe *Curso.java* que representa a tabela *courses* no banco de dados e está representada no código 1.1.

<sup>1</sup> HTTP - Hypertext Transfer Protocol

<sup>2</sup> JPA - Java Persistence API

<sup>3</sup> POJO - Plain Old Java Object

**Código 1.1 – Classe Curso. Fonte:** Elaborado pelos autores.

```
1  @Entity(name = "cursos")
2  public class Curso {
3
4      private Long idCurso;
5      private String nome;
6      private String professor;
7
8      @Id
9      @GeneratedValue
10     @Column(name = "id_curso")
11     public Long getIdCurso() {
12         return idCurso;
13     }
14
15     public void setIdCurso(Long idCurso) {
16         this.idCurso = idCurso;
17     }
18
19     @Column(length = 100, nullable = false)
20     public String getNome() {
21         return nome;
22     }
23
24     public void setNome(String nome) {
25         this.nome = nome;
26     }
27
28     @Column(length = 100, nullable = false)
29     public String getProfessor() {
30         return professor;
31     }
32
33     public void setProfessor(String professor) {
34         this.professor = professor;
35     }
36
37     /**
38      * hashCode e Equals
39      */
40 }
```

Foram criadas outras classes *Java* com a mesma finalidade da anterior, porém com pequenas diferenças no que diz respeito à atributos, metodos e anotações. Estas classes representam, de maneira individual, as tabelas no banco de dados. Certos atributos dessas classes tem por finalidade representar as colunas de cada tabela. Já os atributos que armazenam instâncias de outras classes ou até mesmo conjuntos(coleções) de intâncias representam os relacionamentos entre as tabelas. E por fim, para cada classe que representa um entidade, foi necessário implementar os métodos `hashCode` e `equals`, para que estas pudessem facilmente ser comparadas e diferenciadas em relação aos seus valores, haja vista que cada instância destas classes representa um registro no banco de dados.

Em seguida à criação das entidades, foi necessário configurar o arquivo `persistence.xml`

que fica dentro do *classpath* do projeto *Java* ou seja, dentro da mesma pasta onde estão contidos os pacotes do projeto. Esse arquivo é extremamente importante pois, é nele que estão todas as configurações relativas à conexão com o banco de dados, configurações referentes ao Dialeto SQL que vai ser usado para as consultas e configurações referentes ao *persistence unit* que é o conjunto de classes mapeadas para o banco de dados. O arquivo `persistence.xml` está exposto no código 1.2.

**Código 1.2** – `persistence.xml`. **Fonte:** Elaborado pelos autores.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="WsUnivas">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/wsunivas" />
      <property name="javax.persistence.jdbc.user" value="postgres" />
      <property name="javax.persistence.jdbc.password" value="2289cpm22" />
      <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
      <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="update" />
    </properties>
  </persistence-unit>
</persistence>
```

Em seguida à confecção do `persistence.xml` foi criada a classe `JpaUtil` que está representada na 1.3. Esta classe é responsável por criar uma `EntityManagerFactory` que é uma fabrica de instâncias de `EntityManager` que nada mais é que um *persistence unit* ou unidade de persistência. Essa classe têm a responsabilidade de prover um modo de comunicação entre a aplicação e o banco de dados. Porém a classe `JpaUtil` cria uma única instancia de `EntityManagerFactory`, que é responsável por disponibilizar e gerenciar as instancias de `EntityManager` de acordo com a necessidade da aplicação.

**Código 1.3** – Classe JpaUtil. **Fonte:** Elaborado pelos autores.

---

```
1  public class JpaUtil {
2      private static EntityManagerFactory factory;
3
4      static {
5          factory = Persistence.createEntityManagerFactory("
              WsUnivas");
6      }
7
8      public static EntityManager getEntityManager() {
9          return factory.createEntityManager();
10     }
11
12     public static void close() {
13         factory.close();
14     }
15
16 }
```

---

Em seguida a construção das classes que fazem a parte da persistência de dados, foi desenvolvido a parte de disponibilização de serviços *RESTful*, fazendo uso do *framework Jersey*. Foi necessário, portanto fazer a configuração da aplicação para que fazendo uso deste *framework* a classe que representa o primeiro serviço do *webservice*, que é a classe Alunos. Essa classe representa um contexto

## REFERÊNCIAS

FERREIRA, A. B. H. : **Novo Aurélio Século XXI**: o dicionário da língua portuguesa. 3<sup>a</sup>. ed. Rio de Janeiro: Nova Fronteira, 1999.

GUNTHER, H. : **Como Elaborar um Questionário**. 2003. Disponível em: <[http://www.dcoms.unisc.br/portal/upload/com\\_arquivo/como\\_elaborar\\_um\\_questionario.pdf](http://www.dcoms.unisc.br/portal/upload/com_arquivo/como_elaborar_um_questionario.pdf)>. Acesso em: 15 de Abril de 2015.

MARCONI, M. A.; LAKATOS, E. M. : **Técnicas de pesquisas**: planejamento e execução de pesquisas, amostragens e técnicas de pesquisas, elaboração, análise e interpretação de dados. 5<sup>a</sup>. ed. São Paulo: Atlas, 2002.