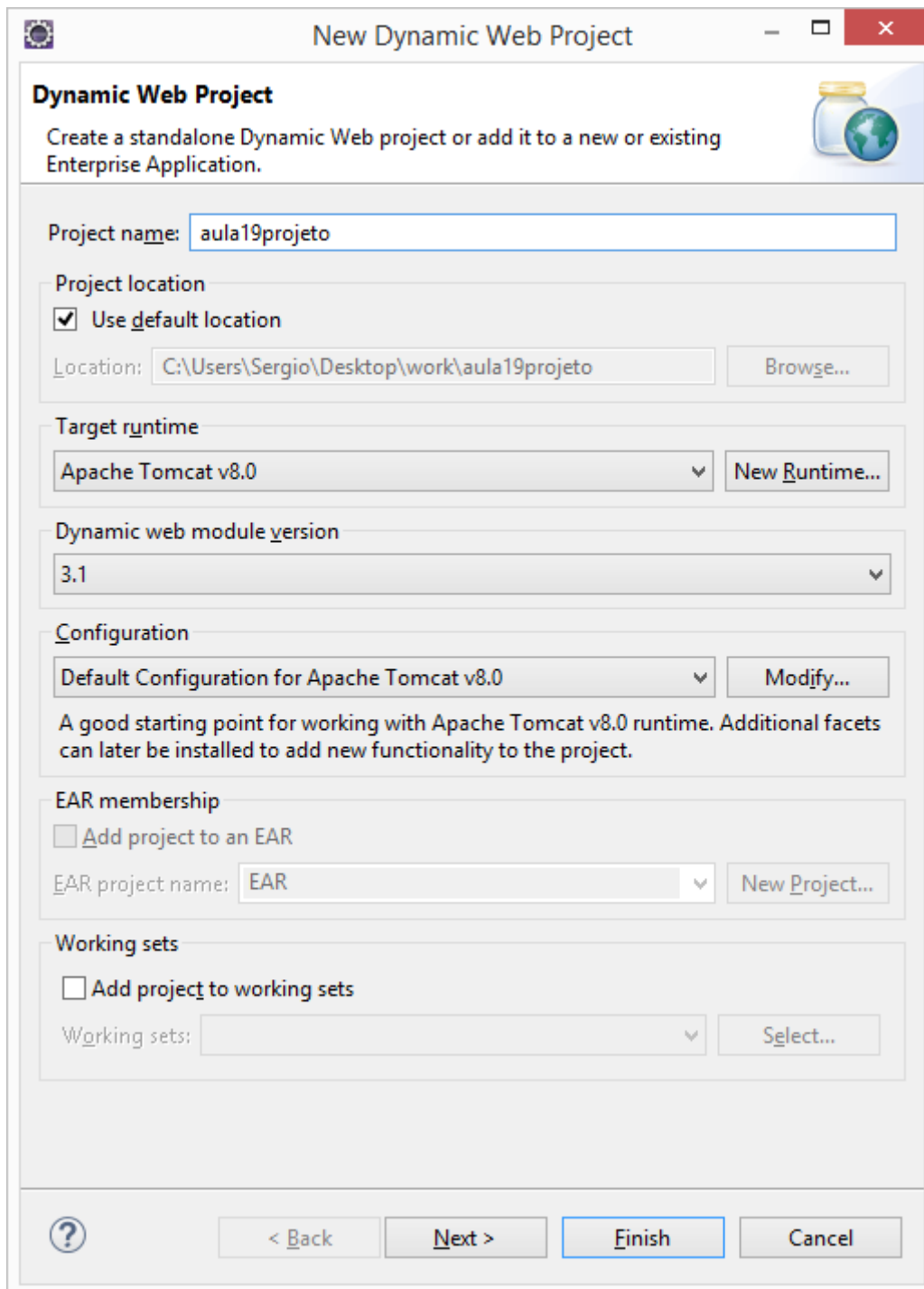


### Criando o projeto:

- File > New > Dynamic Web Project



The screenshot shows the 'New Dynamic Web Project' dialog box in Eclipse. The dialog is titled 'New Dynamic Web Project' and has a subtitle 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' The 'Project name' field is set to 'aula19projeto'. The 'Project location' section has the 'Use default location' checkbox checked, and the 'Location' field shows 'C:\Users\Sergio\Desktop\work\aula19projeto'. The 'Target runtime' is set to 'Apache Tomcat v8.0'. The 'Dynamic web module version' is set to '3.1'. The 'Configuration' section shows 'Default Configuration for Apache Tomcat v8.0'. The 'EAR membership' section has the 'Add project to an EAR' checkbox unchecked, and the 'EAR project name' field is set to 'EAR'. The 'Working sets' section has the 'Add project to working sets' checkbox unchecked, and the 'Working sets' field is empty. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

```
package br.com.brq.entities;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "funcionario")
@NamedQueries(
    {
        @NamedQuery(name = Funcionario.FIND_ALL,
            query = "select f from Funcionario as f")
    }
)
public class Funcionario {

    public static final String FIND_ALL = "funcionario.findall";

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idfuncionario")
    private Integer idFuncionario;

    @Column(name = "nome", length = 50, nullable = false)
    private String nome;

    @Column(name = "salario", nullable = false)
    private Double salario;

    @Temporal(TemporalType.DATE)
    @Column(name = "dataadmissao", nullable = false)
    private Date dataAdmissao;
```

```
public Funcionario() {  
    // TODO Auto-generated constructor stub  
}  
  
public Funcionario(Integer idFuncionario, String nome, Double salario,  
    Date dataAdmissao) {  
    super();  
    this.idFuncionario = idFuncionario;  
    this.nome = nome;  
    this.salario = salario;  
    this.dataAdmissao = dataAdmissao;  
}  
  
public Integer getIdFuncionario() {  
    return idFuncionario;  
}  
  
public void setIdFuncionario(Integer idFuncionario) {  
    this.idFuncionario = idFuncionario;  
}  
  
public String getNome() {  
    return nome;  
}  
  
public void setNome(String nome) {  
    this.nome = nome;  
}  
  
public Double getSalario() {  
    return salario;  
}  
  
public void setSalario(Double salario) {  
    this.salario = salario;  
}  
  
public Date getDataAdmissao() {  
    return dataAdmissao;  
}  
  
public void setDataAdmissao(Date dataAdmissao) {  
    this.dataAdmissao = dataAdmissao;  
}
```

```
@Override
public String toString() {
    return "Funcionario [idFuncionario=" + idFuncionario
        + ", nome=" + nome + ", salario=" + salario
        + ", dataAdmissao=" + dataAdmissao + "];"
}
}
```

---

## Criando o arquivo de configuração para acesso a base de dados: /META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
    version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">

    <!-- nome da configuração mapeada : aula -->
    <persistence-unit name="aula" transaction-type="RESOURCE_LOCAL">

        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <properties>
            <property name="hibernate.dialect"
                value="org.hibernate.dialect.MySQL5InnoDBDialect" />
            <property name="javax.persistence.jdbc.driver"
                value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url"
                value="jdbc:mysql://localhost:3306/aula19" />
            <property name="javax.persistence.jdbc.user"
                value="root" />
            <property name="javax.persistence.jdbc.password"
                value="brqbrq" />

            <property name="hibernate.show_sql"
                value="true" />
            <property name="hibernate.format_sql"
                value="false" />
            <property name="hibernate.use_sql_comments"
                value="false" />
            <property name="hibernate.jdbc.wrap_result_sets"
                value="false" />
            <property name="hibernate.hibernate.cache.use_query_cache"
                value="true" />
            <property name="hibernate.hbm2ddl.auto" value="update" />
        </properties>
    </persistence-unit>
</persistence>
```

## Criando a classe de persistência para Funcionario:

/persistence/DAOFuncionario.java

```
package br.com.brq.persistence;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import br.com.brq.entities.Funcionario;

public class DAOFuncionario {

    //gerenciador de entidades (Hibernate 4)
    private EntityManager manager; //null..

    //construtor..
    public DAOFuncionario() {
        //inicializar o entitymanager..
        manager = Persistence.createEntityManagerFactory("aula")
            .createEntityManager();
    }

    public void insert(Funcionario f) throws Exception{
        manager.getTransaction().begin();
        manager.persist(f); //gravando..
        manager.getTransaction().commit();
    }

    public void update(Funcionario f) throws Exception{
        manager.getTransaction().begin();
        manager.merge(f); //alterando..
        manager.getTransaction().commit();
    }

    public void delete(Funcionario f) throws Exception{
        manager.getTransaction().begin();
        manager.remove(f); //excluindo..
        manager.getTransaction().commit();
    }

    public Funcionario findById(Integer idFuncionario) throws Exception{
        return manager.find(Funcionario.class, idFuncionario);
    }
}
```

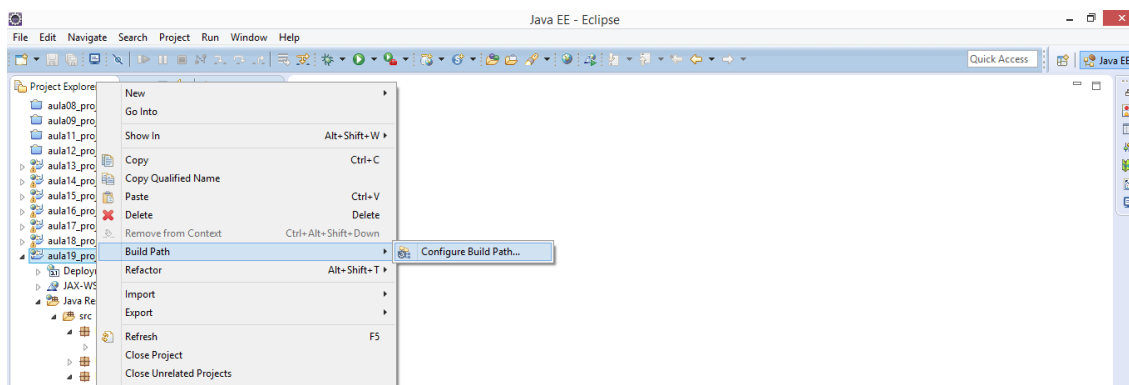
```
@SuppressWarnings("unchecked")
public List<Funcionario> findAll() throws Exception{
    return manager.createNamedQuery(Funcionario.FIND_ALL)
        .getResultList();
}
}
```

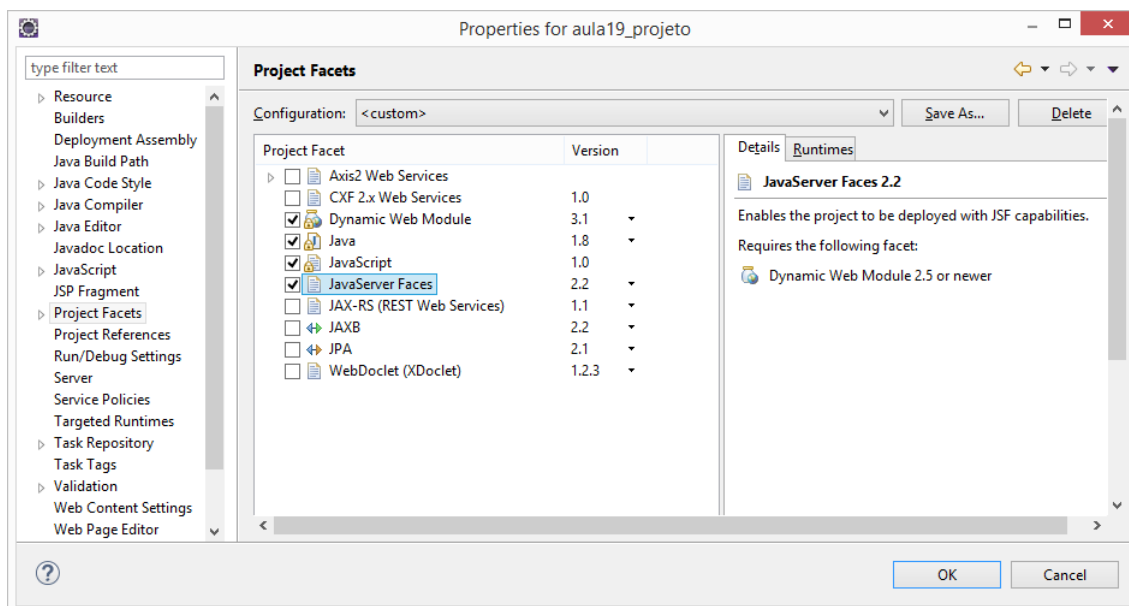
## JSF – Java Server Faces

JavaServer Faces (JSF) é uma especificação Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que o programador se concentre na lógica da aplicação.

- Permite que o desenvolvedor crie UIs através de um conjunto de componentes UIs pré-definidos;
- Fornece um conjunto de tags JSP para acessar os componentes;
- Reutiliza componentes da página;
- Associa os eventos do lado cliente com os manipuladores dos eventos do lado do servidor (os componentes de entrada possuem um valor local representando o estado no lado servidor);
- Fornece separação de funções que envolvem a construção de aplicações Web.
- Utiliza Ajax em alguns de seus componentes tornando alguns processos mais rápidos e eficientes.

## Configurando o JSF no projeto:





### /WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1">
  <display-name>aula19_projeto</display-name>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>redmond</param-value>
  </context-param>
</web-app>
```

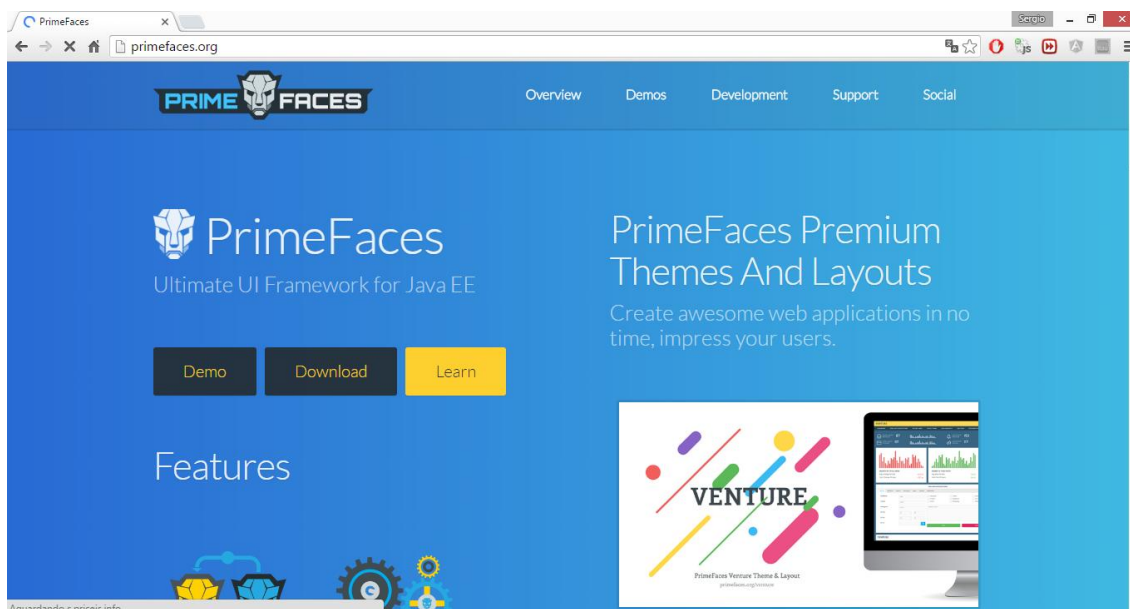
### /WEB-INF/faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
  version="2.2">
</faces-config>
```

## Primefaces

Galeria de components visuais para JSF

<http://primefaces.org/>



Criando a página: **manterfuncionarios.xhtml**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

    <h:head>
        <title>Projeto JSF Hibernate</title>
    </h:head>

    <h:body>

        <p:panel header="Cadastro de Funcionários">
            <h:outputText value="Para cadastrar um funcionario,
                informe os dados abaixo:"/>
            <p:separator/>

            <h:form id="formulario">

                <h:panelGrid columns="2">

                    <p:outputLabel value="Nome do Funcionario:"
                        for="nome"/>
```



```
<p:inputText id="nome" required="true"
    requiredMessage="Por favor, informe o nome do funcionário."
    value="#{mbfunc.funcionario.nome}"/>

<p:outputLabel value="Salário:" for="salario"/>
<p:inputText id="salario" required="true"
    requiredMessage="Por favor, informe o salário do funcionário."
    value="#{mbfunc.funcionario.salario}"/>

<p:outputLabel value="Data de Admissão:" for="dataadmissao"/>
<p:calendar id="dataadmissao" required="true"
    requiredMessage="Por favor, informe a data de admissão do
    funcionário."
    value="#{mbfunc.funcionario.dataAdmissao}"/>

</h:panelGrid>

<p:commandButton value="Cadastrar Funcionário"
    ajax="true" update=":formulario, :formconsulta"
    action="#{mbfunc.cadastrar}"/>

<p:messages/>
<p:growl/>

</h:form>

<p:separator/>

<h:form id="formconsulta">

    <h:outputText value="Relação de Funcionários cadastrados:"/>

    <p:dataTable paginator="true" rows="5" emptyMessage="Não há registros"
        value="#{mbfunc.listagemFuncionarios}" var="f">

        <f:facet name="header">
            <h:outputText value="Consulta de Funcionários"/>
        </f:facet>

        <p:column headerText="Código">
            <h:outputText value="#{f.idFuncionario}"/>
        </p:column>

        <p:column headerText="Nome do Funcionario"
            filterBy="#{f.nome}" filterMatchMode="startsWith">
            <h:outputText value="#{f.nome}"/>
        </p:column>

        <p:column headerText="Salário">
            <h:outputText value="#{f.salario}"/>
        </p:column>

    </p:dataTable>

</h:form>
```

```
<p:column headerText="Data de Admissão">
    <h:outputText value="#{f.dataAdmissao}"/>
</p:column>

</p:dataTable>

</h:form>

</p:panel>

</h:body>

</html>
```

---

### Criando o ManagedBean para gerenciar os eventos da página:

```
package br.com.brq.managedbeans;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;

import br.com.brq.entities.Funcionario;
import br.com.brq.persistence.DAOFuncionario;

@ManagedBean(name = "mbfunc")
@ViewScoped
public class ManagedBeanFuncionario {

    //atributo para resgatar os campos do formulário..
    private Funcionario funcionario;

    //atributo para retornar os funcionarios da base de dados..
    private List<Funcionario> listagemFuncionarios;

    //construtor..
    public ManagedBeanFuncionario() {
        funcionario = new Funcionario(); //instanciando..
    }

    //método para a ação de cadastro do funcionario..
    public void cadastrar(){
```

```
String mensagem = null;
```

```
try{
    DAOFuncionario d = new DAOFuncionario();
    d.insert(funcionario); //gravando..

    mensagem = "Funcionario " + funcionario.getNome()
               + ", cadastrado com sucesso." ;

    funcionario = new Funcionario(); //novo espaço de memória..
}
catch(Exception e){
    mensagem = e.getMessage();
}

//exibir mensagem dentro do formulário de cadastro...
FacesContext.getCurrentInstance()
    .addMessage("formulario", new FacesMessage(mensagem));
}
```

```
public Funcionario getFuncionario() {
    return funcionario;
}
```

```
public void setFuncionario(Funcionario funcionario) {
    this.funcionario = funcionario;
}
```

```
public List<Funcionario> getListagemFuncionarios() {

    try{
        DAOFuncionario d = new DAOFuncionario(); //persistencia..
        listagemFuncionarios = d.findAll(); //executando a consulta..
    }
    catch(Exception e){
        e.printStackTrace();
    }

    return listagemFuncionarios;
}

}
```

### Resultado:

Cadastro de Funcionários

Para cadastrar um funcionario, informe os dados abaixo:

Nome do Funcionario: \*

Salário: \*

Data de Admissão: \*

☒ Por favor, informe o nome do funcionário.  
Por favor, informe o salário do funcionário.  
Por favor, informe a data de admissão do funcionário.

Relação de Funcionários cadastrados:

Código	Nome do Funcionario	Salário	Data de Admissão
1	Sergio Mendes	1000.0	2016-06-08
2	Joao Pedro	2000.0	2016-06-07
3	Ana Paula	3000.0	2016-06-06
4	Ana Maria	4000.0	2016-06-08
5	Carlos da Silva	2000.0	2016-06-07

Cadastro de Funcionários

Para cadastrar um funcionario, informe os dados abaixo:

Nome do Funcionario: \*

Salário: \*

Data de Admissão: \*

Relação de Funcionários cadastrados:

Código	Nome do Funcionario	Salário	Data de Admissão
1	Sergio Mendes	1000.0	2016-06-08
2	Joao Pedro	2000.0	2016-06-07
3	Ana Paula	3000.0	2016-06-06
4	Ana Maria	4000.0	2016-06-08
5	Carlos da Silva	2000.0	2016-06-07