FIAP Fase 1 : Capítulo 4

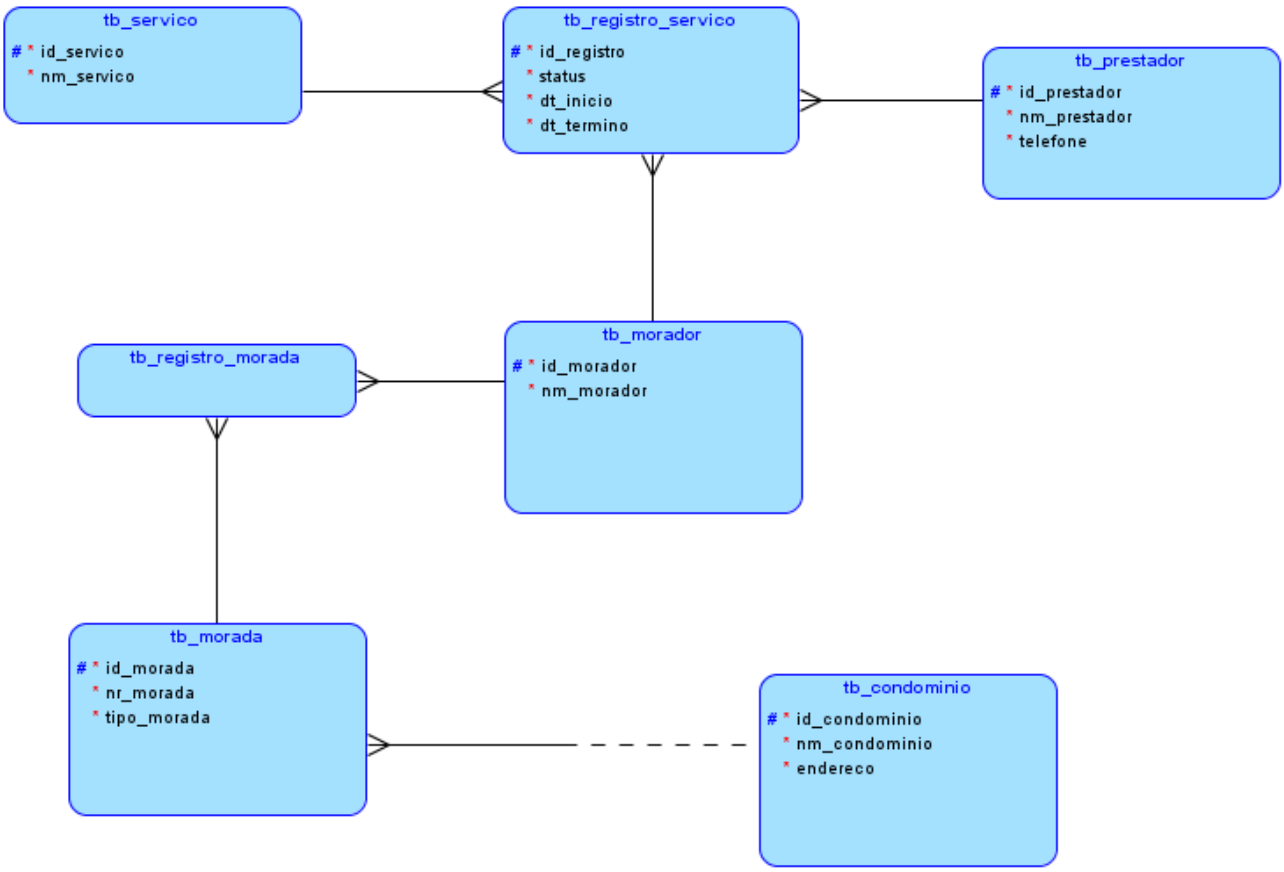
Início

Para abstrair o conhecimento adquirido no capítulo 4 da Fase 1 desenvolvi o seguinte modelo de negócios:

Temos moradores que ocupam um determinado condomínio. Estes podem está procurando serviços domésticos de diversos tipos (elétrico, hidráulico ou mecânico) e o prestador de serviço detém a capacitação.

A ideia é criar uma aplicação que facilite o encontro entre moradores e prestadores de serviço. E caso o morador deseje contratar o serviço haverá a criação de um registro entre as partes.

Segue abaixo a representação gráfica do modelo relacional.



- **tb_condominio:** Tabela que vai armazenar dados de condomínio. Contém o nome e endereço do condomínio.
- **tb_morada:** Tabela que vai armazenar dados de morada. Contém o numero da morada e seu tipo (CASA ou APARTAMENTO).
- **Relação morada_condominio:** Uma morada deve pertencer à um condomínio. Um condomínio pode possuir uma ou mais de uma morada.
- **tb_morador:** Tabela que vai armazenar dados de morador. Contém nome do morador.
- **Relação morada_morador:** Uma morada deve ser habitada por um ou mais moradores e um morador deve morar em pelo menos uma morada.

- **tb_servico:** Tabela que vai armazenar dados do serviço. Contém o nome do serviço.
- **tb_prestador:** Tabela que vai armazenar dados do prestador de serviço. Contém o nome, número de telefone do prestador.
- **tb_registro_servico:** Tabela que vai armazenar dados do registro de serviço. Contém a data de inicio, a data de fim de serviço e o status do registro.
- **Relação servico_registro:** Um tipo de serviço deve ser registrado e um registro deve conter um tipo de serviço.
- **Relação prestador_registro:** Um prestador de serviço deve ser registrado e um registro deve conter um prestador de serviço.
- **Relação morador_registro:** Um morador deve ser registrado e um registro deve conter um morador.

OBJETIVO DO PROJETO

Conforme foi pedido na atividade vamos nos concentrar somente na criação das tabelas entidades **Estabelecimento** (exemplo), **Registro** e **Prestador** por enquanto utilizando o `Hibernate` .

Setup

Primeiramente temos que configurar nosso projeto, segue passo a passo:

- Cria o arquivo maven java
- Configura o arquivo pom.xml da seguinte forma:

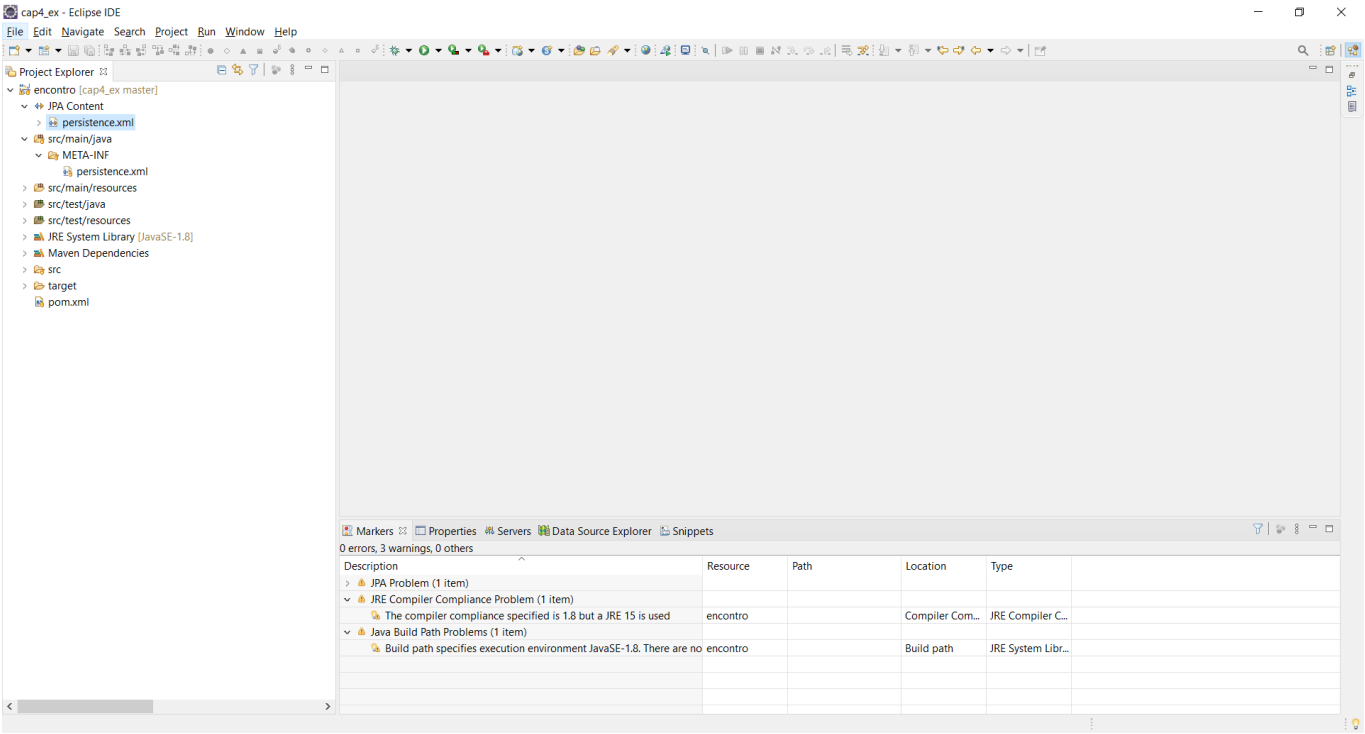
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>br.com.encontro</groupId>
  <artifactId>encontro</artifactId>
  <version>0.0.1</version>
  <dependencies>

    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>RELEASE</version>
    </dependency>

    <dependency>
        <groupId>oracle</groupId>
        <artifactId>jdbc-driver</artifactId>
        <version>12</version>
        <scope>system</scope>
        <systemPath>C:/Users/lucca/Oracle/ojdbc8.jar</systemPath>
    </dependency>

  </dependencies>
```

```
</project>
```

- Converte o arquivo para JPA
- Configura o arquivo persistence.xml inicialmente da seguinte forma:

```
<?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="smartcities-orm" transaction-type="RESOURCE_LOCAL">

        <provider>
            org.hibernate.jpa.HibernatePersistenceProvider
        </provider>

        <properties>
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto" value="create" />
            <property name="hibernate.dialect"
value="org.hibernate.dialect.Oracle12cDialect" />
            <property name="javax.persistence.jdbc.driver"
value="oracle.jdbc.OracleDriver" />
            <property name="javax.persistence.jdbc.url"
value="jdbc:oracle:thin:@oracle.fiap.com.br:1521:ORCL" />
            <property name="javax.persistence.jdbc.user" value="XXXXX" />
            <property name="javax.persistence.jdbc.password" value="XXXXX" />
        </properties>

    </persistence-unit>
</persistence>
```

Feita a configuração inicial!



Criação das entidades

1. **Prestador.java**

```
package br.com.encontro.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

@Entity
@Table(name="tb_prestador")
public class Prestador {

    @Id
    @SequenceGenerator(name="prestador",sequenceName="sq_tb_prestador",alloc
ationSize=1)
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="prestador")
    @Column(name="id_prestador")
    private int id;

    @Column(name="nm_prestador",nullable=false,length=100)
    private String nome;

    @Column(name="nr_morador",nullable=false)
    private int telefone;

    public Prestador() {
```

```
    }

    public Prestador(int id,String nome, int telefone) {
        this.id = id;
        this.nome = nome;
        this.telefone = telefone;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getTelefone() {
        return telefone;
    }

    public void setTelefone(int telefone) {
        this.telefone = telefone;
    }



}
```

Após criada a classe incluir a seguinte linha em `persistence.xml`

```
<class>br.com.encontro.entity.Prestador</class>
```

2. **Registro.java**

```
package br.com.encontro.entity;

import java.util.Calendar;

import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

@Entity
@Table(name="tb_registro_servico")
public class Registro {

    @Id
    @SequenceGenerator(name="registro",sequenceName="sq_tb_registro",allocat
ionSize=1)
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="registro")
    @Column(name="id_registro")
    private int id;

    @Enumerated(EnumType.STRING)
    @Column(name="status")
    private Estado tipo;

    @CreationTimestamp
    @Column(name="dt_data_cadastro")
    private Calendar dataCadastro;

    @UpdateTimestamp
    @Column(name="dt_data_modificacao")
    private Calendar dataModificacao;

    public Registro() {
    }

    public Registro(int id, Estado tipo) {
        this.id = id;
        this.tipo = tipo;
    }

    public Calendar getDataCadastro() {
        return dataCadastro;
    }

    public void setDataCadastro(Calendar dataCadastro) {
        this.dataCadastro = dataCadastro;
    }

    public Calendar getDataModificacao() {
        return dataModificacao;
    }
```

```
    public void setDataModificacao(Calendar dataModificacao) {
        this.dataModificacao = dataModificacao;
    }



}

enum Estado {

    ABERTO, FECHADO

}
```

Após criada a classe incluir a seguinte linha em `persistence.xml`

```
<class>br.com.encontro.entity.Registro</class>
```

3. **Estabelecimento.java**

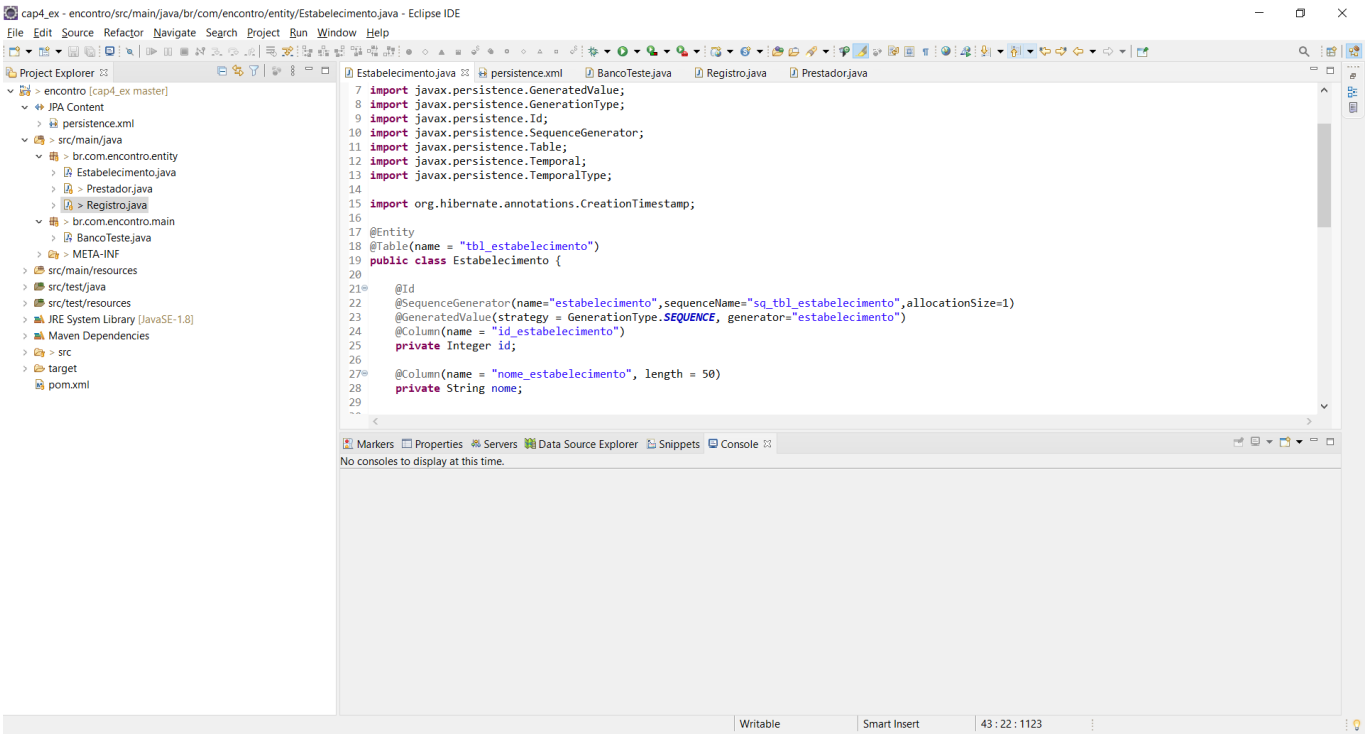
```
package br.com.encontro.entity;

import java.util.Calendar;

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

import org.hibernate.annotations.CreationTimestamp;

@Entity
@Table(name = "tbl_estabelecimento")
public class Estabelecimento {

    @Id

@SequenceGenerator(name="estabelecimento",sequenceName="sq_tbl_estabelecimen
to",allocationSize=1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
generator="estabelecimento")
```

```
    @Column(name = "id_estabelecimento")
    private Integer id;

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


    @CreationTimestamp
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "dh_criacao")
    private Calendar dataCriacao;


    public Integer getId() {
        return id;
    }


    public void setId(Integer id) {
        this.id = id;
    }


    public String getNome() {
        return nome;
    }


    public void setNome(String nome) {
        this.nome = nome;
    }


    public Calendar getDataCriacao() {
        return dataCriacao;
    }


    public void setDataCriacao(Calendar dataCriacao) {
        this.dataCriacao = dataCriacao;
    }


}
```

Após criada a classe incluir a seguinte linha em `persistence.xml`

```
<class>br.com.encontro.entity.Estabelecimento</class>
```

Pronto! Todas as entidades foram criadas com sucesso!

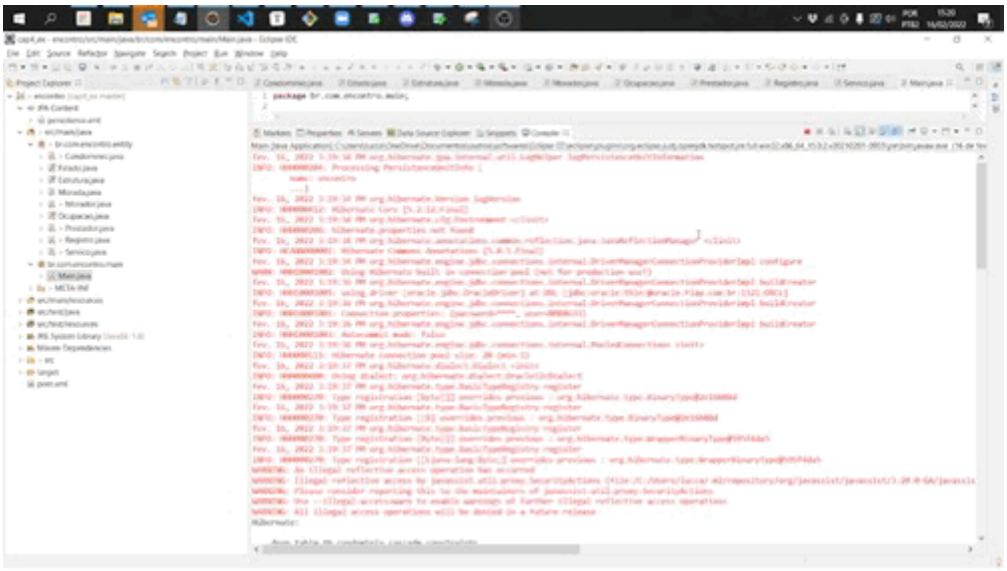


Realizando testes

Agora vamos testar se o programa está criando nossas tabelas de forma correta. Contudo antes de tudo devemos criar a classe `BancoTeste.java` que vai apresentar o método main().

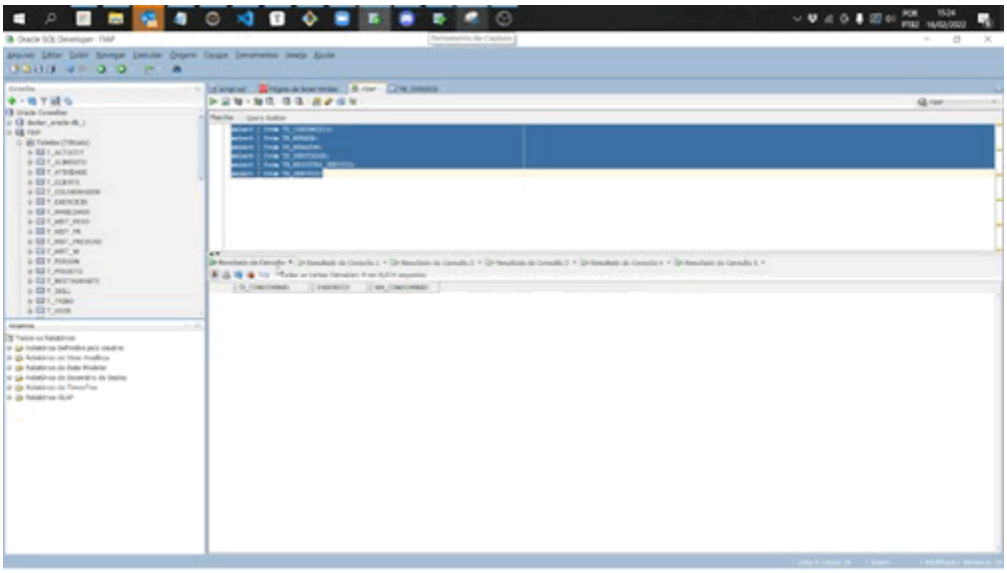
```
package br.com.encontro.main;

import javax.persistence.Persistence;

public class BancoTeste {

    public static void main(String[] args) {
        Persistence.createEntityManagerFactory("smartcities-
orm").createEntityManager();

    }

}
```

Vamos aos testes!



Como pode ser visto acima as tabelas foram criadas com sucesso! (Pode ser que ocorra erros nos drops de tabelas, sequences e constraints se os elementos forem criados pela primeira vez, contudo nada que vá afetar a criação das entidades).

Conferindo no banco de dados Oracle, podemos notar a presença das novas tabelas.



CLIQUE AQUI PARA VISUALIZAR O PROJETO NO GITHUB

FORTE ABRAÇO!