

DEFENSA HITO4

DIEGO EMILIANO RIVERA TAPIA

CREAMOS UN PROYECTO EN EL FRAMEWORK SPRING

The screenshot shows the Spring Initializr web application at start.spring.io. A yellow arrow points from the left towards the 'Project' section, labeled 'Datos del proyecto'. Another yellow arrow points upwards from the bottom towards the 'Dependencies' section, labeled 'Añadimos las dependencias'.

Project

- Maven Project
- Gradle Project

Language

- Java
- Kotlin
- Groovy

Spring Boot

- 2.4.0 (SNAPSHOT)
- 2.3.2 (SNAPSHOT)
- 2.3.1
- 2.2.9 (SNAPSHOT)
- 2.2.8
- 2.1.16 (SNAPSHOT)
- 2.1.15

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	<input checked="" type="radio"/> Jar

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

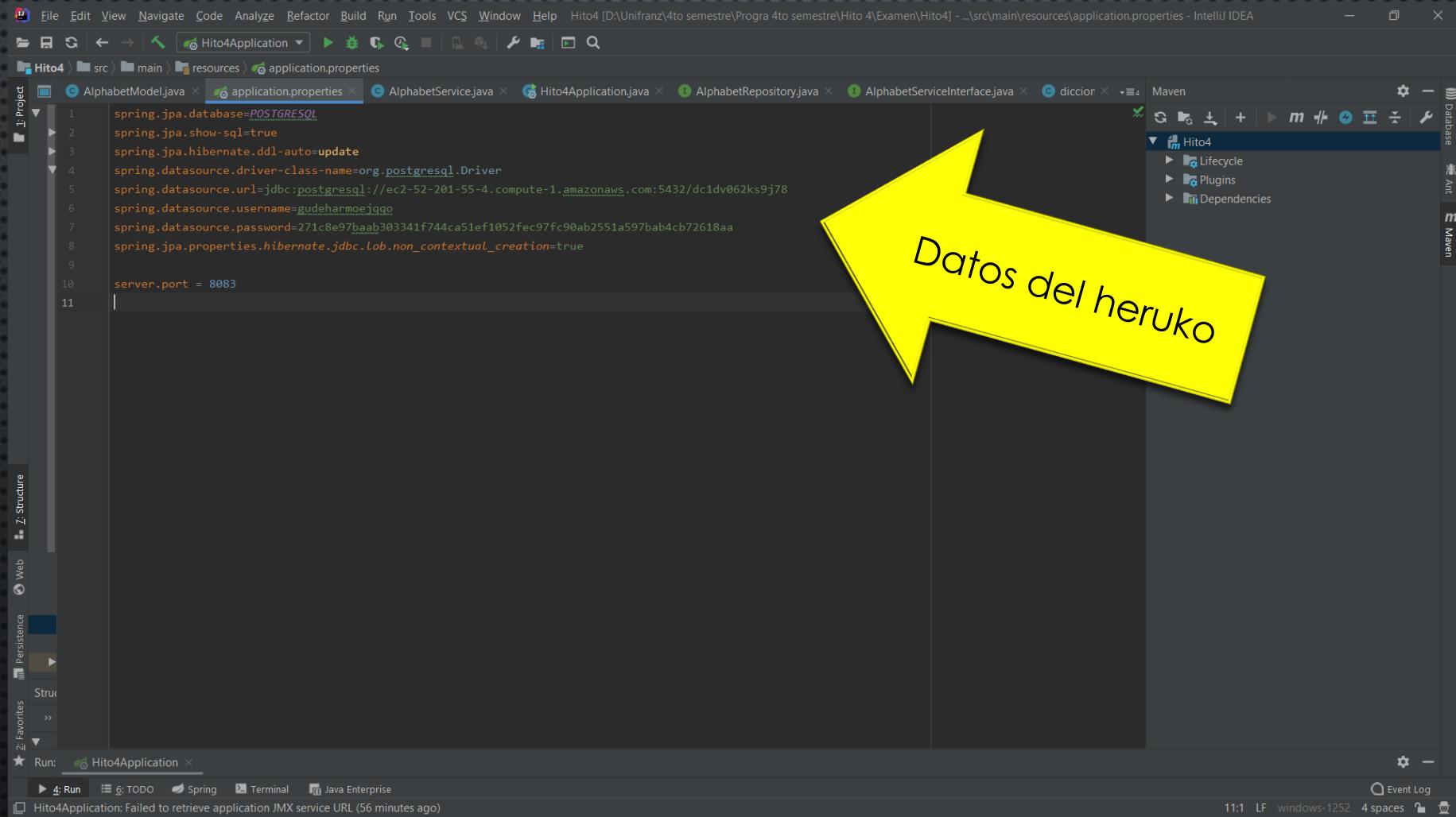
Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

PostgreSQL Driver SQL
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Buttons

- GENERATE CTRL + F
- EXPLORE CTRL + SPACE
- SHARE...

CREAMOS LA CONEXIÓN CON HERUKO Y LA BASE DE DATOS



The screenshot shows the IntelliJ IDEA interface with the project 'Hito4' open. The 'application.properties' file is selected in the left-hand navigation bar. The code editor displays the following configuration:

```
spring.jpa.database=POSTGRESQL
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://ec2-52-201-55-4.compute-1.amazonaws.com:5432/dc1dv062ks9j78
spring.datasource.username=gudeharmoniqoo
spring.datasource.password=271c8e97baab303341f744ca51ef1052fec97fc90ab2551a597bab4cb72618aa
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true

server.port = 8083
```

A large yellow arrow points from the text "Datos del heruko" to the database URL in the code. The right-hand side of the interface shows the Maven repository browser with the 'Hito4' project selected.

Datos del heruko

MODELO: CREAMOS LAS TABLAS PARA LAS LETRAS

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Hito4 [D:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4] - ..\src\main\java\com\defensa\Hito4\Model\AlphabetModel.java  
Hito4 src main java com defensa Hito4 Model AlphabetModel  
Project Pyt 1  
1 package com.defensa.Hito4.Model;  
2  
3 import javax.persistence.*;  
4  
5 @Entity  
6 @Table(name = "AlphabetTest")  
7 public class AlphabetModel {  
8     @Id  
9     @GeneratedValue(strategy = GenerationType.IDENTITY)  
10    private Integer id;  
11  
12    @Column(name = "letter", length = 100, nullable = false)  
13    private String letter;  
14  
15    @Column(name = "tipeR", length = 100, nullable = false)  
16    private String tipeR;  
17  
18    public AlphabetModel(){  
19    }  
20  
21    public AlphabetModel(String letter, String tipeR){  
22        this.letter = letter;  
23        this.tipeR = tipeR;  
24    }  
25  
26    public Integer getId() { return id; }  
27  
28    public void setId(Integer id) { this.id = id; }  
29  
30    public String getLetter() { return letter; }  
31  
32    public void setLetter(String letter) { this.letter = letter; }  
33  
34  
35  
36  
37  
38  
39  
40  
41  
AlphabetModel  
Run: Hito4Application  
Run TODO Spring Terminal Java Enterprise  
Hito4Application: Failed to retrieve application JMX service URL (20 minutes ago)  
41:2 CRLF UTF-8 4 spaces Event Log
```

Damos un nombre a la tabla

Creamos las columnas

Getters and Setters

ALPHABETREPOSITORY

CREAMOS EL REPOSITORIO PARA LAS LETRAS

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Hito4 [D:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4] - ...\\main\\java\\com\\defensa\\Hito4\\Repository\\AlphabetRepository.java - X
Hito4 Application > Hito4 > Repository > AlphabetRepository.java
Hito4Application.java AlphabetModel.java AlphabetService.java AlphabetRepository.java diccionarioModel.java Hito4iml UtilJava diccionari Maven
Project Structure Database Maven Artifacts
Hito4 Lifecycle Plugins Dependencies
Maven
1 package com.defensa.Hito4.Repository;
2
3
4 import com.defensa.Hito4.Model.AlphabetModel;
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.data.jpa.repository.Query;
7 import org.springframework.stereotype.Repository;
8
9 import java.util.List;
10
11 @Repository
12 public interface AlphabetRepository extends JpaRepository<AlphabetModel, Integer> {
13     @Query( value = "SELECT * FROM Alphabet WHERE tipoR='First'", nativeQuery = true)
14     public List<AlphabetModel> getFirstRow();
15
16     @Query( value = "SELECT * FROM Alphabet WHERE tipoR='Second'", nativeQuery = true)
17     public List<AlphabetModel> getSecondRow();
18
19     @Query( value = "SELECT * FROM Alphabet WHERE tipoR='Three'", nativeQuery = true)
20     public List<AlphabetModel> getThreeRow();
21 }
22
```

Creamos un repository

Obtenemos las letras por filas

ALPHABETSERVICEINTERFACE

CREAMOS UNA INTERFAZ PARA NUESTRO SERVICIO

The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project Bar:** Hito4Application, Hito4, src, main, java, com, defensa, Hito4, Service, AlphabetServiceInterface.java.
- Code Editor:** The current file is `AlphabetServiceInterface.java`, which contains the following code:

```
package com.defensa.Hito4.Service;

import com.defensa.Hito4.Model.AlphabetModel;
import java.util.List;

public interface AlphabetServiceInterface {
    public void saveData();
    public List<AlphabetModel> getAllLettersFirst();
    public List<AlphabetModel> getAllLettersSecond();
    public List<AlphabetModel> getAllLettersThree();
}
```
- Toolbars:** Standard toolbar with icons for Run, Stop, Refresh, etc.
- Sidebar:** Shows the project structure, Maven dependencies, and other settings.
- Status Bar:** Shows the time (14:1), encoding (CRLF, UTF-8), and code style (4 spaces).

ALPHABET SERVICE

- CREAMOS EL SERVICE PARA LAS LETRAS, Y GUARDAR LOS DATOS EN LA BASE DE DATOS

The screenshot shows an IDE interface with a Java file named `AlphabetService.java` open. The code implements the `AlphabetServiceInterface`. It contains static final strings for letter groups (`Q_P`, `A_L`, `Z_M`) and an `ArrayList` for saving data. It includes methods to save data and retrieve lists of letters. Three yellow arrows point from the right side of the slide to the code, each accompanied by a Spanish annotation:

- A yellow arrow points to the declaration of the letter groups: `private static final String Q_P = "Q,M,E,R,T,Y,U,I,O,P";`, `private static final String A_L = "A,S,D,F,G,H,J,K,L";`, and `private static final String Z_M = "Z,X,C,V,B,N,M";`. The annotation is: **Creamos las variables de las letras**.
- A yellow arrow points to the `saveData()` method, which uses the `alphabetRepository` to save three letter models with specific types: `First`, `Second`, and `Three`. The annotation is: **Guardamos las letras en la base de datos**.
- A yellow arrow points to the three overridden methods: `getAllLettersFirst()`, `getAllLettersSecond()`, and `getAllLettersThree()`, which return lists of `AlphabetModel` objects. The annotation is: **Creamos los métodos**.

```
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class AlphabetService implements AlphabetServiceInterface {
    private static final String Q_P = "Q,M,E,R,T,Y,U,I,O,P";
    private static final String A_L = "A,S,D,F,G,H,J,K,L";
    private static final String Z_M = "Z,X,C,V,B,N,M";

    @Autowired
    private AlphabetRepository alphabetRepository;

    @Override
    public void saveData() {
        if(alphabetRepository.count()==0){
            alphabetRepository.save(new AlphabetModel(Q_P, tipeR: "First"));
            alphabetRepository.save(new AlphabetModel(A_L, tipeR: "Second"));
            alphabetRepository.save(new AlphabetModel(Z_M, tipeR: "Three"));
        }
    }

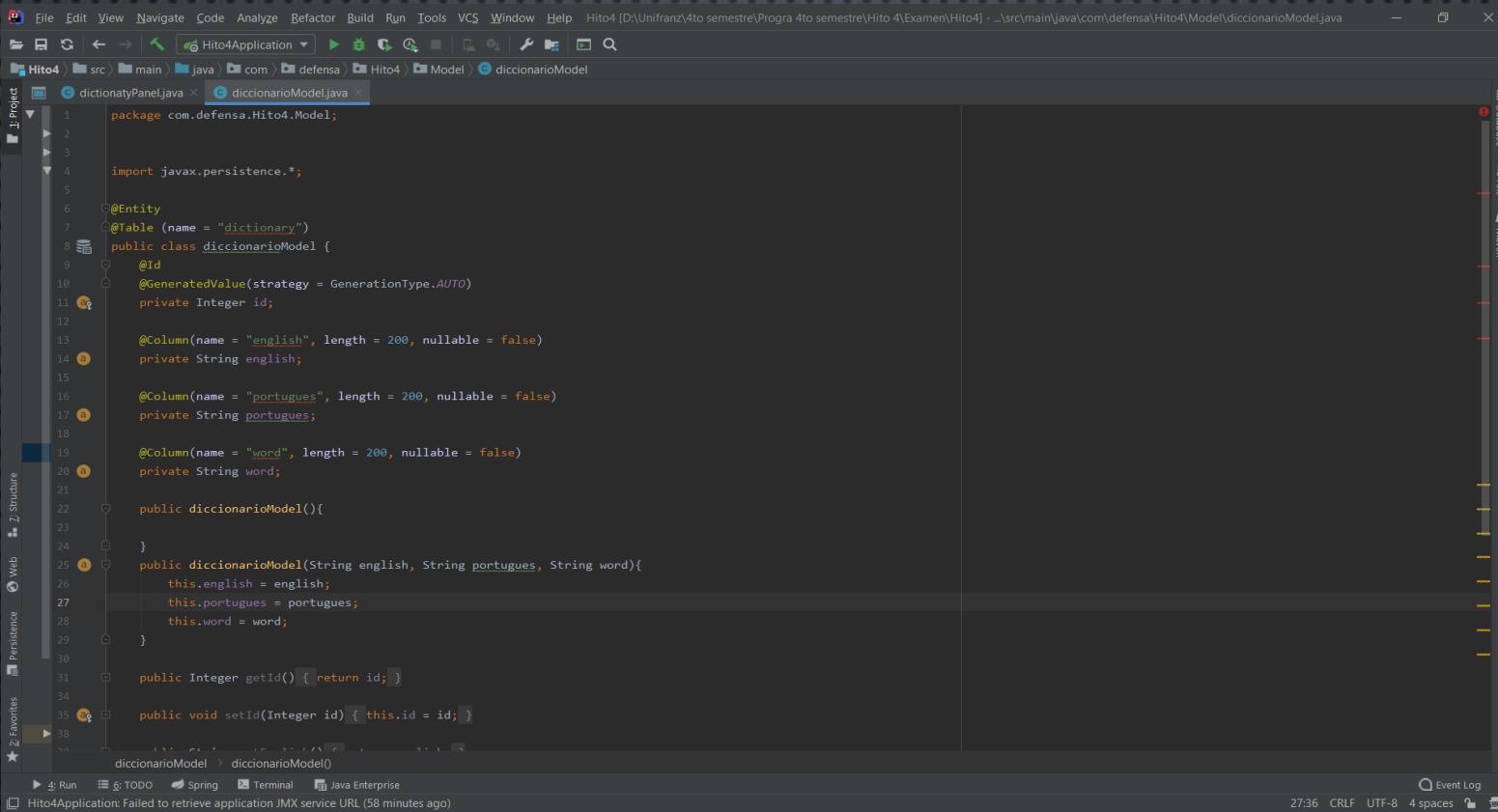
    @Override
    public List<AlphabetModel> getAllLettersFirst() { return alphabetRepository.getFirstRow(); }

    @Override
    public List<AlphabetModel> getAllLettersSecond() { return alphabetRepository.getSecondRow(); }

    @Override
    public List<AlphabetModel> getAllLettersThree() { return alphabetRepository.getThreeRow(); }
}
```

CREAMOS EL MODELO PARA EL DICCIONARIO

- SEGUIMOS LOS MISMO PASOS QUE PARA LA TABLA LETRAS



The screenshot shows an IDE interface with the following details:

- Title Bar:** Hito4 [D:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4] - ...src\main\java\com\defensa\Hito4\Model\diccionarioModel.java
- Project Tree:** Shows the project structure with files like dictionaryPanel.java and diccionarioModel.java.
- Code Editor:** Displays the Java code for the diccionarioModel class. The code defines an entity with three columns: id, english, and portugues, and a word attribute.

```
1 package com.defensa.Hito4.Model;
2
3 import javax.persistence.*;
4
5 @Entity
6 @Table (name = "dictionary")
7 public class diccionarioModel {
8     @Id
9     @GeneratedValue(strategy = GenerationType.AUTO)
10    private Integer id;
11
12    @Column(name = "english", length = 200, nullable = false)
13    private String english;
14
15    @Column(name = "portugues", length = 200, nullable = false)
16    private String portugues;
17
18    @Column(name = "word", length = 200, nullable = false)
19    private String word;
20
21    public diccionarioModel(){
22    }
23
24    public diccionarioModel(String english, String portugues, String word){
25        this.english = english;
26        this.portugues = portugues;
27        this.word = word;
28    }
29
30    public Integer getId() { return id; }
31
32    public void setId(Integer id) { this.id = id; }
33
34 }
```

- Toolbars and Status Bar:** Standard IDE toolbars and status bar at the bottom.

CREAMOS EL REPOSITORIO PARA DICCIONARIO

The screenshot shows a Java code editor within an IDE. The project is named "Hito4" and the current file is "diccionarioRepository.java". The code defines a repository interface for a "diccionarioModel" entity:

```
package com.defensa.Hito4.Repository;

import com.defensa.Hito4.Model.diccionarioModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface diccionarioRepository extends JpaRepository<diccionarioModel, Integer> {
    @Query(value = "SELECT * FROM dictionary WHERE word = :wordSelected", nativeQuery = true)
    public diccionarioModel getWordTranslate(@Param("wordSelected") String wordSelected);
}
```

The IDE's navigation bar shows other files like "diccionarioPanel.java" and "diccionarioModel.java". The bottom status bar indicates the application failed to retrieve JMX service URL.

CREAMOS EL INTERFAZ DEL SERVICIO DICCIONARIO

The screenshot shows an IDE interface with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The title bar indicates the project is "Hito4" and the file open is "diccionarioServiceInterface.java". The left sidebar displays the project structure under "Project":

- com.defensa.Hito4
 - Gui
 - Frame
 - MainFrame
 - Listener
 - ButtonLister
 - Panels
 - diccionaryP
 - Util
 - Util
 - Model
 - AlphabetMode
 - diccionarioMo
 - Repository
 - AlphabetRepo
 - diccionarioRe
 - Service
 - AlphabetServ
 - AlphabetServ
 - diccionarioSer
 - diccionarioSer
 - Hito4Application
 - resources
 - static
 - templates
 - application.properties
- Structure
- Web
- Persistence
- Favorites
- External Libraries
- Scratches and Consoles

The code editor shows the following Java code:

```
package com.defensa.Hito4.Service;  
  
public interface diccionarioServiceInterface {  
    public void saveData();  
    String traducir(String t, String l);  
}
```

A large yellow arrow points from the text "Métodos" to the code in the editor.

The status bar at the bottom right shows "8:1 CRLF UTF-8 4 spaces" and an "Event Log" icon.

CREAMOS EL SERVICIO PARA EL DICCIONARIO

```
@Service
public class diccionarioService implements diccionarioServiceInterface {
    @Autowired
    private diccionarioRepository dictionaryRepository;
    @Override
    public void saveData() {
        if (dictionaryRepository.count() == 0) {
            dictionaryRepository.save(new diccionarioModel(english: "MONDAY", portuguese: "SEGUNDA-FEIRA", word: "LUNES"));
            dictionaryRepository.save(new diccionarioModel(english: "TUESDAY", portuguese: "TERCA-FEIRA", word: "MARTES"));
            dictionaryRepository.save(new diccionarioModel(english: "WEDNESDAY", portuguese: "QUARTA-FEIRA", word: "MIERCOLES"));
            dictionaryRepository.save(new diccionarioModel(english: "THURSDAY", portuguese: "QUINTA-FEIRA", word: "JUEVES"));
            dictionaryRepository.save(new diccionarioModel(english: "FRIDAY", portuguese: "SEXTA-FEIRA", word: "VIERNES"));
            dictionaryRepository.save(new diccionarioModel(english: "SATURDAY", portuguese: "SABADO", word: "SABADO")));
            dictionaryRepository.save(new diccionarioModel(english: "SUNDAY", portuguese: "DOMINGO", word: "DOMINGO")));
        }
    }
    @Override
    public String traducir(String t, String l) {
        diccionarioModel dictionaryModel = dictionaryRepository.getWordTranslate(t);
        String ingles = dictionaryModel.getEnglish();
        String ln_i = "INGLES";
        String word = dictionaryModel.getWord();
        String ln_w = "ESPAÑOL";
        String portuguese = dictionaryModel.getPortugues();
        String ln_p = "PORTUGUES";
        String traducción = "";
        if(l.equals(ln_i)){
            traducción = ingles;
        }
        if(l.equals(ln_w)){
            traducción = word;
        }
        if(l.equals(ln_p)){
            traducción = portuguese;
        }
        return traducción;
    }
}
```

Guardamos en la base de datos

```
}
```

```
    @Override
    public String traducir(String t, String l) {
        diccionarioModel dictionaryModel = dictionaryRepository.getWordTranslate(t);
        String ingles = dictionaryModel.getEnglish();
        String ln_i = "INGLES";
        String word = dictionaryModel.getWord();
        String ln_w = "ESPAÑOL";
        String portuguese = dictionaryModel.getPortugues();
        String ln_p = "PORTUGUES";
        String traducción = "";
        if(l.equals(ln_i)){
            traducción = ingles;
        }
        if(l.equals(ln_w)){
            traducción = word;
        }
        if(l.equals(ln_p)){
            traducción = portuguese;
        }
        return traducción;
    }
}
```

Método para traducir

CREAMOS EL DISEÑO DE LOS PANELES

```
1. Project: Hito4
2. Editor: dictionaryPanel.java
3. Tools: Ratón, Texto, Dibujar, Estampar, Flecha, Borrador, Formato, Deshacer, Rehacer, Borrar, Guardar
4. Status Bar: C:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4 - ...\\src\\main\\java\\com\\defensa\\Hito4\\Gui\\Panels\\dictionaryPanel.java
5. Right Panel: Database, Ant, Maven
```

```
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
```

```
@Component
public class dictionaryPanel extends JPanel {
    @Autowired
    private AlphabetService alphabetService;

    @Autowired
    private diccionarioService diccionarioService;

    JLabel label = new JLabel();
    JTextField word = new JTextField();
    JLabel label2 = new JLabel();
    JTextField language = new JTextField();
    JLabel label3 = new JLabel();
    JTextField result = new JTextField();

    public dictionaryPanel(){
        System.setProperty("butBackColor", "#C1ECF1");
        System.setProperty("panelcolor", "#EEEEEE");
        System.setProperty("textColor", "#B0B0F6");
        this.setPreferredSize(new Dimension( width: 750, height: 400));
        this.setBackground(Color.getColor("panelcolor"));
        this.setLayout(new GridLayout( rows: 5, cols: 0));
    }
    @PostConstruct
    public void createButtonsLetters(){
        java.util.List<AlphabetModel> firstRow = alphabetService.getAllLettersFirst();
        this.add(this.createPanelButton(firstRow.get(0).getLetter().split( regex: "\\")));
        java.util.List<AlphabetModel> secondRow = alphabetService.getAllLettersSecond();
        String[] titleAlphabet1 = secondRow.get(0).getLetter().split( regex: "\\");
        JPanel panelA_L = this.createPanelButton(titleAlphabet1);
        this.add(panelA_L);
    }
}
```

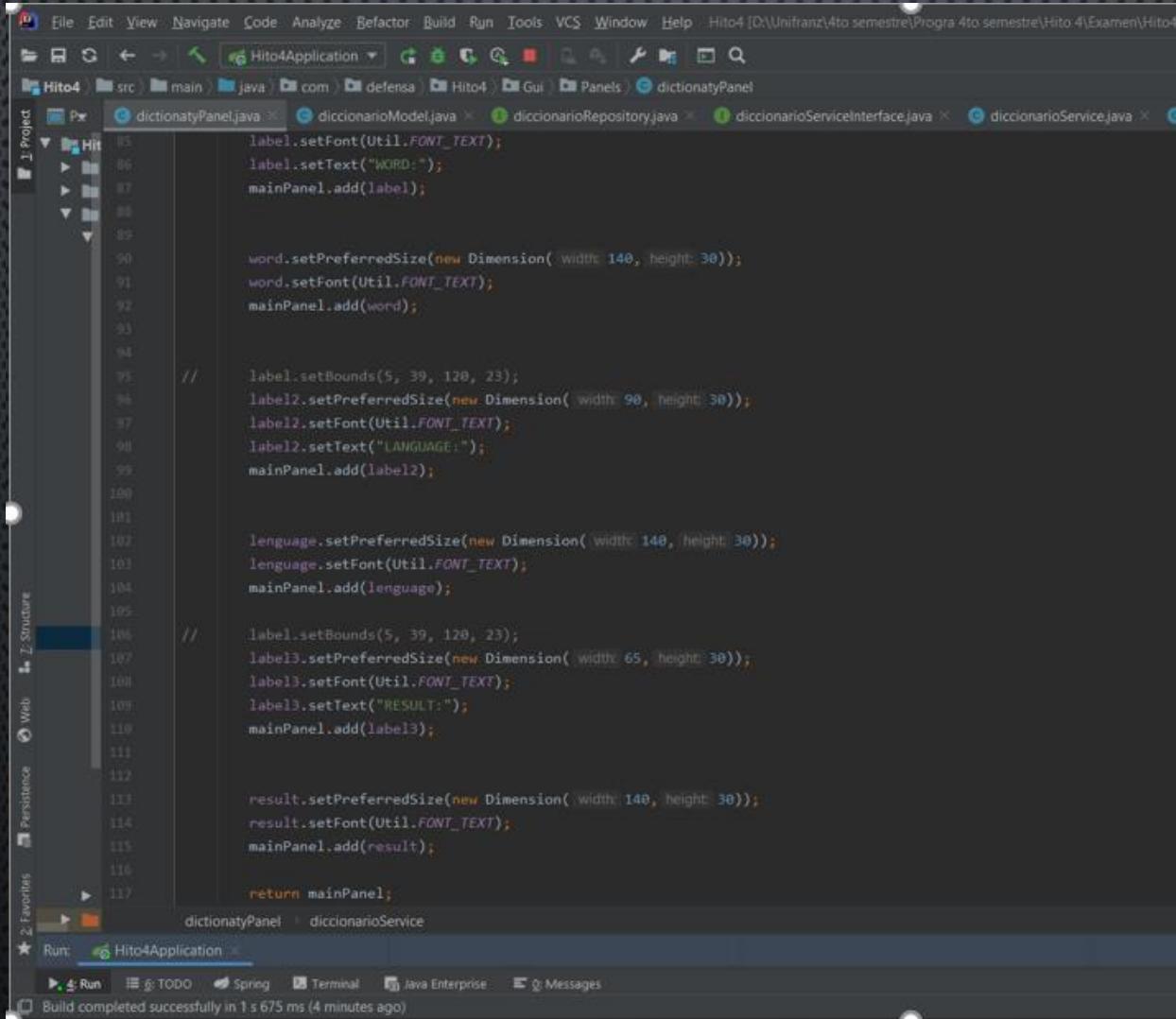
Declaramos los txt Globalmente

Características del panel

Un post Constructor

CREAMOS UN PANEL PARA LAS LETRAS

CONTINUA CREANDO LOS LABEL ETC



The screenshot shows a Java code editor within an IDE. The project is named "Hito4". The current file is "dictionaryPanel.java". The code defines a method that creates several UI components, specifically labels, and adds them to a main panel.

```
label.setFont(Util.FONT_TEXT);
label.setText("WORD:");
mainPanel.add(label);

word.setPreferredSize(new Dimension( width: 140, height: 30));
word.setFont(Util.FONT_TEXT);
mainPanel.add(word);

// label.setBounds(5, 39, 120, 23);
label2.setPreferredSize(new Dimension( width: 90, height: 30));
label2.setFont(Util.FONT_TEXT);
label2.setText("LANGUAGE:");
mainPanel.add(label2);

language.setPreferredSize(new Dimension( width: 140, height: 30));
language.setFont(Util.FONT_TEXT);
mainPanel.add(language);

// label.setBounds(5, 39, 120, 23);
label3.setPreferredSize(new Dimension( width: 65, height: 30));
label3.setFont(Util.FONT_TEXT);
label3.setText("RESULT:");
mainPanel.add(label3);

result.setPreferredSize(new Dimension( width: 140, height: 30));
result.setFont(Util.FONT_TEXT);
mainPanel.add(result);

return mainPanel;
```

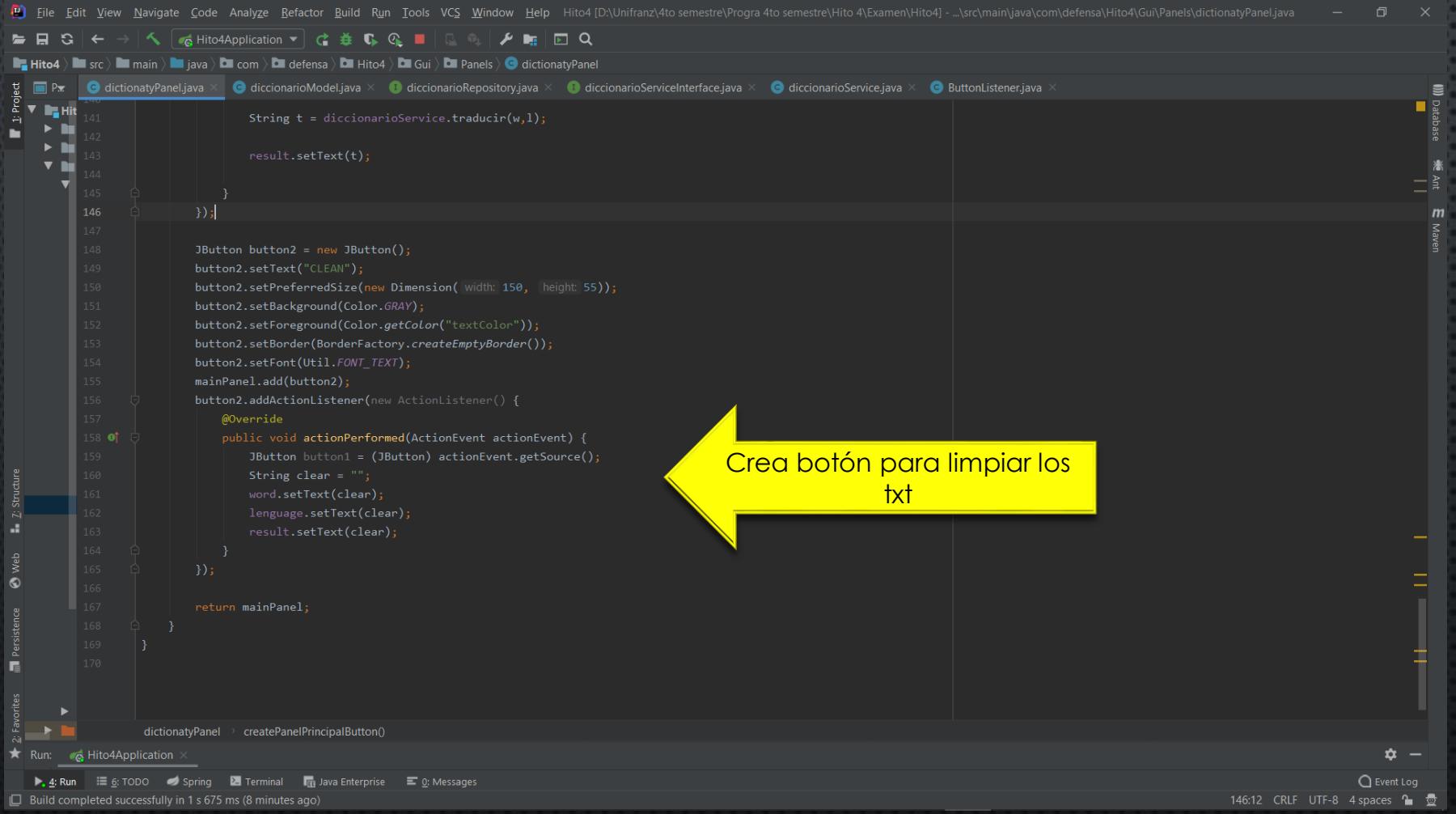
The code uses a dark theme and includes imports for Util.FONT_TEXT, Dimension, and various class names like dictionaryModel, diccionarioRepository, diccionarioServiceInterface, and diccionarioService.

CREA LOS PANELES DE BOTONES

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Hito4 [D:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4] - ...src\main\java\com\defensa\Hito4\Gui\Panels\dictionaryPanel.java  
Hito4 > src > main > java > com > defensa > Hito4 > Gui > Panels > dictionaryPanel.java  
Project I-Project Hit 118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
dictionaryPanel.java x diccionarioModel.java x diccionarioRepository.java x diccionarioServiceInterface.java x diccionarioService.java x ButtonListener.java x  
public JPanel createPanelPrincipalButton(){  
    JPanel mainPanel = new JPanel();  
    mainPanel.setLayout(new FlowLayout());  
    ButtonListener listener = new ButtonListener();  
    JButton button = new JButton();  
    button.setPreferredSize(new Dimension( width: 150, height: 55));  
    //button.addActionListener(listener);  
    button.setText("TRANSLATE");  
    button.setBackground(Color.getColor("butBackColor"));  
    button.setForeground(Color.getColor("textColor"));  
    button.setBorder(BorderFactory.createEmptyBorder());  
    button.setFont(Util.FONT_TEXT);  
    mainPanel.add(button);  
  
    button.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent actionEvent) {  
            JButton button2 = (JButton) actionEvent.getSource();  
            String w = word.getText();  
            String l = language.getText();  
  
            String t = diccionarioService.traducir(w,l);  
  
            result.setText(t);  
        }  
    });  
  
    JButton button2 = new JButton();  
    button2.setText("CLEAN");  
    button2.setPreferredSize(new Dimension( width: 150, height: 55));  
    dictionaryPanel > createPanelPrincipalButton()  
Run: Hito4Application x  
Run TODO Spring Terminal Java Enterprise Messages  
Build completed successfully in 1 s 675 ms (6 minutes ago)  
146:12 CRLF UTF-8 4 spaces Event Log
```

Crea botón para traducir

Action listener para agarrar los datos y llamar al metodo



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Hito4 [D:\Unifranz\4to semestre\Progra 4to semestre\Hito 4\Examen\Hito4] - ...src\main\java\com\defensa\Hito4\Gui\Panels\diccionaryPanel.java

Hito4 > src > main > java > com > defensa > Hito4 > Gui > Panels > diccionaryPanel

```
String t = diccionarioService.traducir(w,1);
result.setText(t);

JButton button2 = new JButton();
button2.setText("CLEAN");
button2.setPreferredSize(new Dimension( width: 150, height: 55));
button2.setBackground(Color.GRAY);
button2.setForeground(Color.getColor("textColor"));
button2.setBorder(BorderFactory.createEmptyBorder());
button2.setFont(Util.FONT_TEXT);
mainPanel.add(button2);
button2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        JButton button1 = (JButton) actionEvent.getSource();
        String clear = "";
        word.setText(clear);
        lenguage.setText(clear);
        result.setText(clear);
    }
});
return mainPanel;
}
```

Run: Hito4Application

Build completed successfully in 1 s 675 ms (8 minutes ago)

146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170

diccionaryPanel > createPanelPrincipalButton()

Event Log

Crea botón para limpiar los
txt

EJECUCIÓN



END