

# PROCESUAL HITO 4

Nicolas Gonzalo Aguilar Arimoza

4 semestre

24/02/2020

# OBJETIVO DE TRABAJO

- Se tiene como objetivo generar una aplicación de escritorio, en donde esta APP tendrá la capacidad de traducir palabras en ESPANOL al idioma INGLES o PORTUGUES.
- En la primera fase (SPRINT 1) se implementara una prueba de concepto, es decir solo deberá de mostrar traducciones de los días de la semana.
- EJEM: Si el usuario escribe LUNES. (La aplicación deberá de mostrar su traducción en el idioma seleccionado (INGLES - PORTUGUES) )
- Parte TECNICA:
- Crear esta app usando Spring framework y Swing(JAVA).
- Utilizar una base de datos relacional PostgreSQL.
- La base de datos debe estar alojada en HEROKU.



# PRIMERA PREGUNTA

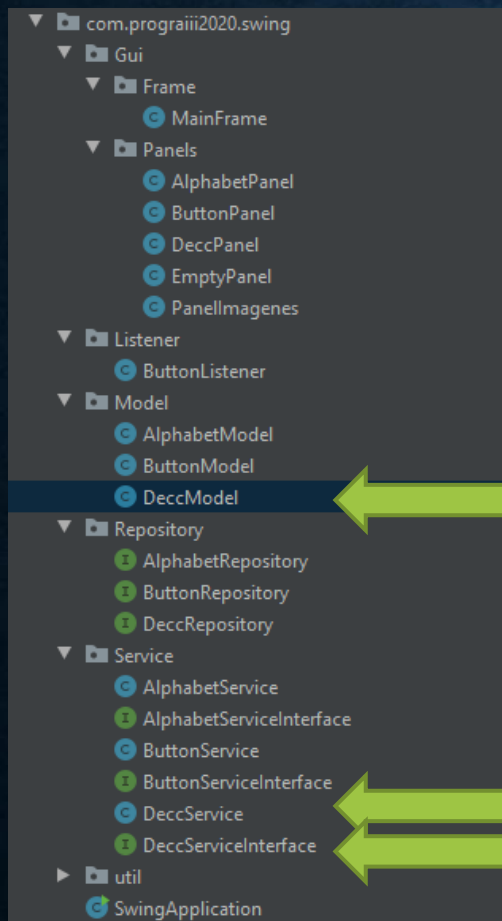
- Realizar la inserción de los datos de los días de la semana en diferentes idiomas

| dictionary |              |  |  |  |
|------------|--------------|--|--|--|
| id         | integer      |  |  |  |
| english    | varchar(200) |  |  |  |
| portugues  | varchar(200) |  |  |  |
| word       | varchar(200) |  |  |  |

|   | id  | english   | portugues     | word      |
|---|-----|-----------|---------------|-----------|
| 1 | 111 | MONDAY    | SEGUNDA-FEIRA | LUNES     |
| 2 | 112 | TUESDAY   | TERCA-FEIRA   | MARTES    |
| 3 | 113 | WEDNESDAY | QUARTA-FEIRA  | MIERCOLES |
| 4 | 114 | THURSDAY  | QUINTA-FEIRA  | JUEVES    |
| 5 | 115 | FRIDAY    | SEXTA-FEIRA   | VIERNES   |
| 6 | 116 | SATURDAY  | SABADO        | SABADO    |
| 7 | 117 | SUNDAY    | DOMINGO       | DOMINGO   |

# PRIMERO SE CREARA LAS TABLAS CON SUS RESPECTIVAS COLUMNAS



Creamos la clase  
Deccmodel para  
la creación de las  
tablas

Insertamos las  
columnas 1 por 1  
dando que serian  
ingles portugués  
Word(español)

Creamos la función  
que recibirá los 3  
parámetros para la  
inserción de la  
tabla

Como también la  
creación  
Deccservice para  
la inserción a las  
tablas

También  
creamos get and  
set de cada uno

```
package com.prograiii2020.swing.modelo;

import javax.persistence.*;

@Entity
@Table(name = "Diccionario")

public class DeccModel {

    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer Id;

    @Column(name = "INGLES", length = 200, nullable = false)
    private String ingles;

    @Column(name = "PORTUGUES", length = 200, nullable = false)
    private String portugues;

    @Column(name = "WORD", length = 200, nullable = false)
    private String word;

    public DeccModel(String ingles, String portugues, String word) {
        this.ingles = ingles;
        this.portugues = portugues;
        this.word = word;
    }

    public Integer getId() { return Id; }

    public void setId(Integer id) { Id = id; }

    public String getIngles() { return ingles; }

    public void setIngles(String ingles) { this.ingles = ingles; }

    public String getPortugues() { return portugues; }

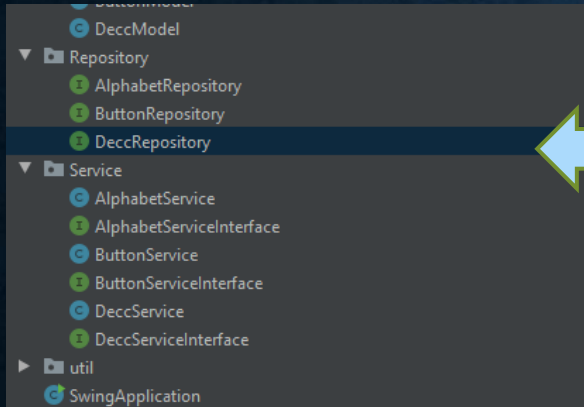
    public void setPortugues(String portugues) { this.portugues = portugues; }

    public String getWord() { return word; }

    public void setWord(String word) { this.word = word; }
}
```



# CREAMOS LA INTERFACE DECCREPOSITORY



Creamos la clase para lograr la conexión de DeccModel y la clase DeccService

```
package com.prograiii2020.swing.Repository;

import com.prograiii2020.swing.Model.ButtonModel;
import com.prograiii2020.swing.Model.DeccModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

public interface DeccRepository extends JpaRepository<DeccModel, Integer> {

}
```

Realizamos la conexión para obtener los datos de la función DeccModel

# CREAMOS LAS CLASES SERVICIO Y SU INTERFACES PARA LOGRAR MANDAR LOS DATOS

Creamos la clase InterfaceService para el guardado de los datos

```
package com.prograiii2020.swing.Service;

import ...

public interface DeccServiceInterface {
    public void saveData();
    public String getRow();
    public List<DeccModel> getDecc();
}
```

Creamos los datos para lograr insertar en la tabla en forma horizontal

Colocaremos los diferentes idiomas español, ingles y portugués

```
@Service
public class DeccService implements DeccServiceInterface {

    @Autowired
    private DeccRepository deccRepository;

    @Override
    public void saveData() {
        if (deccRepository.count() == 0) {
            deccRepository.save(new DeccModel( ingles: "MONDAY",   portugues: "SEGUNDA-FEIRA", word: "LUNES"));
            deccRepository.save(new DeccModel( ingles: "TUESDAY",  portugues: "TERÇA-FEIRA", word: "MARTES"));
            deccRepository.save(new DeccModel( ingles: "WEDNESDAY", portugues: "QUARTA-FEIRA", word: "MIÉRCOLES"));
            deccRepository.save(new DeccModel( ingles: "THURSDAY",  portugues: "QUINTA-FEIRA", word: "JUEVES"));
            deccRepository.save(new DeccModel( ingles: "FRIDAY",    portugues: "SEXTA-FEIRA", word: "VIERNES"));
            deccRepository.save(new DeccModel( ingles: "SATURDAY",  portugues: "SABADO", word: "SABADO"));
            deccRepository.save(new DeccModel( ingles: "SUNDAY",    portugues: "DOMINGO", word: "DOMINGO"));
        }
    }

    @Override
    public String getRow() { return null; }

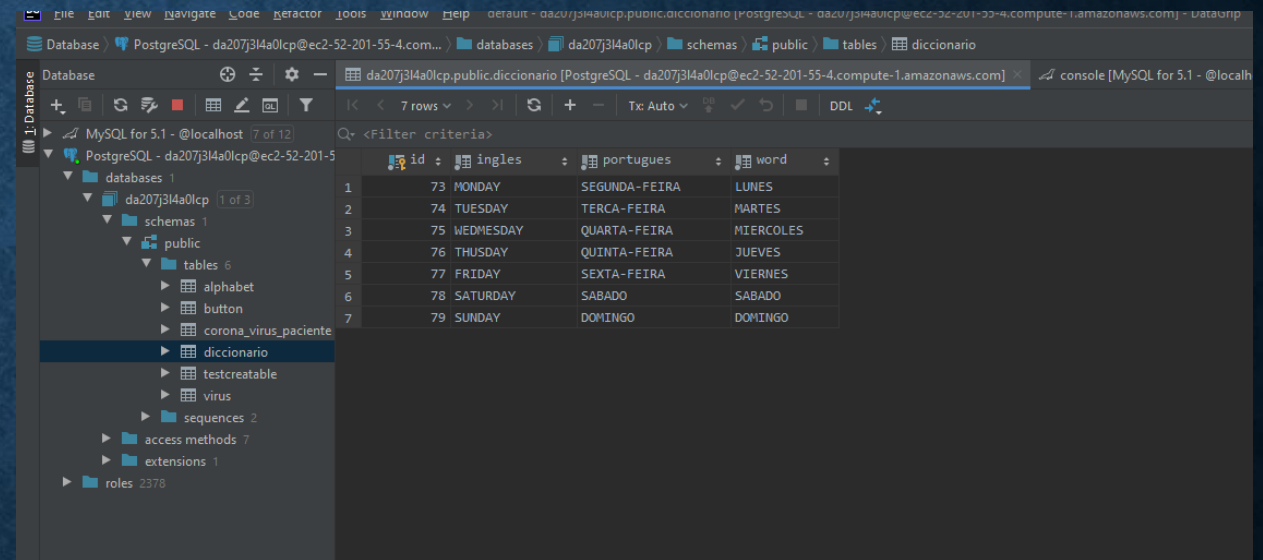
    @Override
    public List<DeccModel> getDecc() {
        return null;
    }
}
```

Utilizamos el deccRepository para lograr guardar las variables en los datos de Deccmodel

# GUARDADO EN EL DATAGRIP

Como resultado se guardara en la base de datos tal y como ingresamos los datos en las en DeccService

Cada uno en idioma correspondiente como también la traducción de cada día a su lado





# PREGUNTA 2

- Realizar el modelo de diseño utilizando código

Q W E R T Y U I O P

A S D F G H J K L

Z X C V B N M

WORD:  LANGUAGE:  RESULT:

TRANSLATE CLEAN



# PARA LA CREACIÓN DEL TECLADO CREAMOS OTRA TABLA EN LA BASE DE DATOS

Creación de las columnas como la tabla

Creación de la función para recibir los datos

Creación de los get y set de cada l

Creación de la clase AlphabetModel

```
import javax.persistence.*;

@Entity
@Table(name = "Alphabet")
public class AlphabetModel {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer Id;

    @Column(name = "letter", length = 200, nullable = false)
    private String letter;

    @Column(name = "typer", length = 10, nullable = false)
    private String typer;

    public AlphabetModel() {

    }

    public AlphabetModel(String letter, String typer) {
        this.letter = letter;
        this.typer = typer;
    }

    public String getTyper() { return typer; }

    public void setTyper(String typer) { this.typer = typer; }

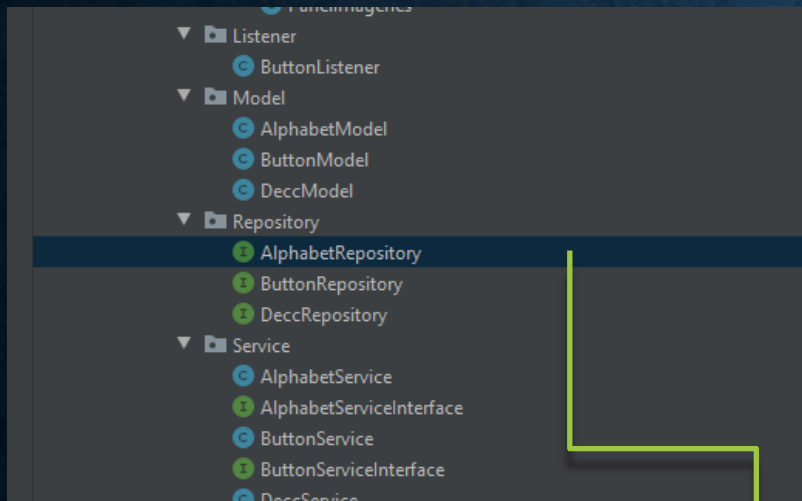
    public Integer getId() { return Id; }

    public void setId(Integer id) { Id = id; }

    public String getLetter() { return letter; }

    public void setLetter(String letter) { this.letter = letter; }
```

# CREACIÓN DE LA INTERFACE ALPHABETREPOSITORY



Utilizamos consultas de la base de datos para lograr obtener los datos de cada línea

```
package com.prograii2020.swing.Repository;

import ...

public interface AlphabetRepository extends JpaRepository<AlphabetModel, Integer> {

    @Query(value = "select * from alphabet where typer = 'first';", nativeQuery = true)
    public List<AlphabetModel> getFirstRow();

    @Query(value = "select * from alphabet where typer = 'second';", nativeQuery = true)
    public List<AlphabetModel> getSecondRow();

    @Query(value = "select * from alphabet where typer = 'three';", nativeQuery = true)
    public List<AlphabetModel> getThreeRow();
}
```

Creación de la interface AlphabetRepository



# CREACIÓN DE LA CLASE ALPHABETSERVICE Y LA INTERFACE INTERFACESERVICE

```
1 package com.prograiii2020.swing.Service;
2
3 import ...
4
5
6
7 public interface AlphabetServiceInterface {
8     public void saveData();
9     public List<AlphabetModel> getAllLettersFirst();
10    public List<AlphabetModel> getAllLettersSecond();
11    public List<AlphabetModel> getAllLettersThree();
12 }
13
```

Creamos un get para cada uno y lograr ordenarlos de forma correcta

Creamos los datos para cada fila que ingresaremos en la tabla

Mandamos los datos para lograr guardarlos por fila

Con los gets creados mandamos cada fila

```
package com.prograiii2020.swing.Service;
import ...
@Service
public class AlphabetService implements AlphabetServiceInterface{
    private static final String Q_P = "Q,W,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, type: "first"));
            alphabetRepository.save(new AlphabetModel(A_L, type: "second"));
            alphabetRepository.save(new AlphabetModel(Z_M, type: "three"));
        }
    }

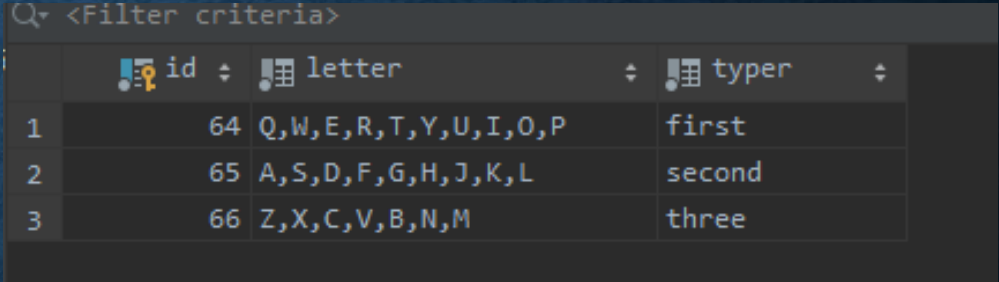
    @Override
    public List<AlphabetModel> getAllLettersFirst() { return alphabetRepository.getFirstRow(); }

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

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

# GUARDADO EN EL DATAGRIP

- Al finalizar se guarda cada dato en fila
- El typer nos ayudara a separar cada línea
- Al momento de obtener los datos
- Como vimos en las consultas del repository

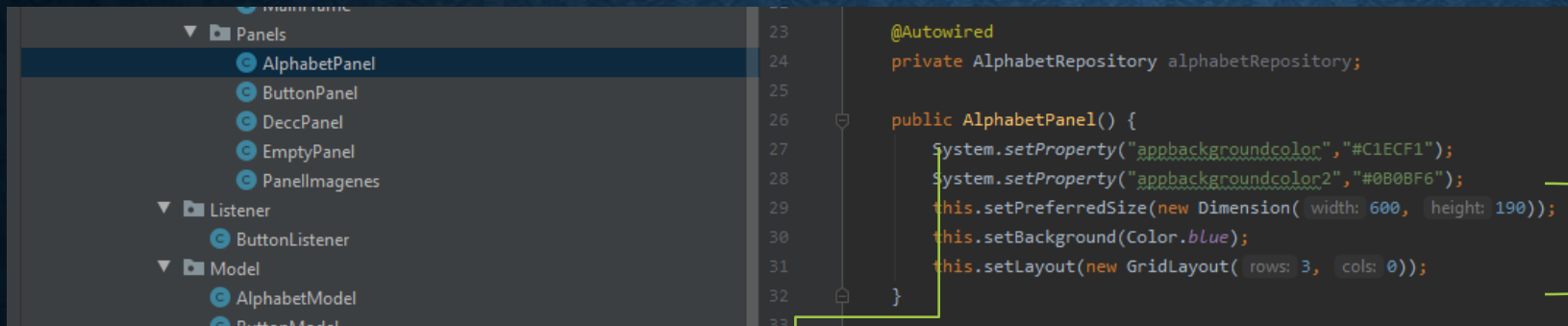


Q+ <Filter criteria>

|   | id | letter              | typer  |
|---|----|---------------------|--------|
| 1 | 64 | Q,W,E,R,T,Y,U,I,O,P | first  |
| 2 | 65 | A,S,D,F,G,H,J,K,L   | second |
| 3 | 66 | Z,X,C,V,B,N,M       | three  |



# CREACION DEL ALPHABETPANEL



Damos las dimensiones y colores del panel como también en que línea queremos que este del programa

Colores que podemos seleccionar mediante código

# CREACIÓN DE LAS TECLAS

```
@PostConstruct
public void createButtonsLetters() {
    List<AlphabetModel> firstRow = alphabetService.getAllLettersFirst();
    String[] titleAlphabet = firstRow.get(0).getLetter().split( regex: "," );
    JPanel panelQ_P = this.createPanelButton(titleAlphabet);
    this.add(panelQ_P);

    List<AlphabetModel> secondRow = alphabetService.getAllLettersSecond();
    String[] titleAlphabet1 = secondRow.get(0).getLetter().split( regex: "," );
    JPanel panelA_L = this.createPanelButton(titleAlphabet1);
    this.add(panelA_L);

    List<AlphabetModel> threeRow = alphabetService.getAllLettersThree();
    String[] titleAlphabet2 = threeRow.get(0).getLetter().split( regex: "," );
    JPanel panelZ_M = this.createPanelButton(titleAlphabet2);
    this.add(panelZ_M);
}
```

Controlamos la primera línea de la base de datos con la función list

Llamamos el getfirst del Alphabetservice para obtener los datos

Controlamos las separaciones con el split

El Split leerá toda la lista y cada vez que encuentre una , separara cada letra para guardar en cuadrados

Al lograr la separación creara el botón con la función createButton



```
public JPanel createPanelButton(String[] titleAlphabet){  
    JPanel mainPanel = new JPanel();  
    mainPanel.setLayout(new FlowLayout());  
    ButtonListener listener = new ButtonListener();  
  
    for(String title : titleAlphabet){  
        JButton button = new JButton(title);  
        button.setPreferredSize(new Dimension( width: 55, height: 40));  
        button.addActionListener(listener );  
        button.setBackground(Color.getColor("appbackgroundcolor"));  
        button.setForeground(Color.getColor("appbackgroundcolor2"));  
        button.setBorder(BorderFactory.createEmptyBorder());  
        button.setFont(util.font_text);  
        mainPanel.add(button);  
    }  
    return mainPanel;  
}
```

La función obtendrá un dato que  
será la letra para poder crear su  
botón

Damos las dimensiones de cada  
botón

Damos los colores que queremos q  
obtengan

Añadimos al panel ya  
creado

# CREACIÓN DEL PANEL DECC

```
public class DeccPanel extends JPanel {
```

```
    public DeccPanel(){
        this.setPreferredSize(new Dimension( width: 900, height: 100));
        this.setLayout(new GridLayout( rows: 1, cols: 0));
    }
```

```
    @PostConstruct
```

```
    public void createButtonsLabel() {
        JPanel caja1 = this.botonas();
        this.add(caja1);
    }
```

Creamos el panel  
dando las  
dimensiones y  
columnas

Creamos los textos de  
cada uno dando las  
dimensiones

También se creara los textos  
para ingresar la palabra que  
queramos introducir de igual  
forma damos sus dimensiones

Creamos los botones de traducir y  
limpiar para la traducción y lograr  
borrar lo que ya se ingreso

Label  
Text  
Label  
Tex  
Label  
Text  
Button  
button

```
public JPanel botones()
{
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new FlowLayout());

    JLabel la1 = new JLabel( text: "word- ");
    la1.setPreferredSize(new Dimension( width: 70, height: 30));
    JLabel la2 = new JLabel( text: "language- ");
    la2.setPreferredSize(new Dimension( width: 70, height: 30));
    JLabel la3 = new JLabel( text: "result- ");
    la3.setPreferredSize(new Dimension( width: 70, height: 30));

    JTextField tex1 = new JTextField();
    tex1.setPreferredSize(new Dimension( width: 70, height: 20));
    JTextField tex2 = new JTextField();
    tex2.setPreferredSize(new Dimension( width: 70, height: 20));
    JTextField tex3 = new JTextField();
    tex3.setPreferredSize(new Dimension( width: 70, height: 20));

    JButton button = new JButton( text: "transformar");
    button.setPreferredSize(new Dimension( width: 150, height: 50));

    JButton button2 = new JButton( text: "limpiar");
    button2.setPreferredSize(new Dimension( width: 150, height: 50));

    mainPanel.add(la1);
    mainPanel.add(tex1);
    mainPanel.add(la2);
    mainPanel.add(tex2);
    mainPanel.add(la3);
    mainPanel.add(tex3);
    mainPanel.add(button);
    mainPanel.add(button2);

    return mainPanel;
}
```

para terminar  
colocamos en  
orden para que  
lo coloque en el  
panel



# DISEÑO TERMINADO

- Al terminar el diseño se creara la imagen con los botones de igual forma los textos para lograr introducir y también los botones de traducir y limpiar

Q W E R T Y U I O P

A S D F G H J K L

Z X C V B N M

WORD:  LANGUAGE:  RESULT:

TRANSLATE CLEAN

# PREGUNTA 3

- Hacer que el programa traduzca la palabra ingresada

A screenshot of a web application interface for translating words. The interface is light gray and contains three input fields at the top: 'WORD: LUNES', 'LANGUAGE: INGLES', and 'RESULT: MONDAY'. The 'RESULT' field is highlighted with a yellow border. Below the input fields are two buttons: a light blue 'TRANSLATE' button and a gray 'CLEAN' button. Two yellow arrows point to the 'LUNES' and 'INGLES' input fields.

| Field    | Value  |
|----------|--------|
| WORD     | LUNES  |
| LANGUAGE | INGLES |
| RESULT   | MONDAY |

Buttons: TRANSLATE, CLEAN



# CREACIÓN DE LA VARIABLE TRADUCIR

```
import ...
```

```
public interface DeccServiceInterface {  
    public void saveData();  
    String traducir(String t, String l);  
    public String getRow();  
    public List<DeccModel> getDecc();  
}
```

La creamos para  
lograr obtener los  
datos de los tex  
box creados en el  
panel

Mandamos los textos para que  
busque en la base de datos y  
reconozca a que idioma  
quiere cambiar

Retorna el texto traducido  
para completar el programa

```
@Override  
public String traducir(String t, String l) {  
    DeccModel deccMode = deccRepository.getWordTranslate(t);  
    String ingles = deccMode.getInglés();  
    String ln_i = "INGLES";  
    String word = deccMode.getWord();  
    String ln_w = "ESPAÑOL";  
    String portugues = deccMode.getPortugues();  
    String ln_p = "PORTUGUES";  
    String traduccion = "";  
    if(l.equals(ln_i)){  
        traduccion = ingles;  
    }  
    if(l.equals(ln_w)){  
        traduccion = word;  
    }  
    if(l.equals(ln_p)){  
        traduccion = portugues;  
    }  
    return traduccion;  
}
```

# CONSULTA EN EL DECCREPOSITORY

```
package com.prograiii2020.swing.Repository;

import com.prograiii2020.swing.Model.ButtonModel;
import com.prograiii2020.swing.Model.DeccModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

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

Creamos la consulta para que obtenga todos los datos de traductor para lograr la respuesta

Lo mandamos al DeccModel para que lo revise si esta correctamente ingresado




# CREACIÓN DE LAS FUNCIONES DE LOS BOTONES

```
@Override
public void actionPerformed(ActionEvent actionEvent) {
    JButton button = (JButton) actionEvent.getSource();
    String w = tex1.getText();
    String l = tex2.getText();

    String t = deccService.traducir(w,l);

    tex3.setText(t);
}
```



Mandamos los text al DeccService para lograr la traducción y logre analizar a que idioma queremos convertir

Al finalizar retornara la respuesta traducida de la palabra que se esperaba

```
button2.setPreferredSize(new Dimension( width: 150, height: 50));
button2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        JButton button2 = (JButton) actionEvent.getSource();
        String clear = "";
        tex1.setText(clear);
        tex2.setText(clear);
        tex3.setText(clear);
    }
});
```

Limpiara todo lo que se halla ingresado en los text para permitir colocar otras palabras para la traduccion

# FINALIZACION DE PROYECTO

Al finalizar el proyecto logramos obtener la traducción de todos los días en los idiomas ingles portugués y viceversa



The screenshot shows a web application window titled "Defensa Hito4 Progra III". The interface features a virtual keyboard with three rows of light blue buttons containing the letters Q, W, E, R, T, Y, U, I, O, P; A, S, D, F, G, H, J, K, L; and Z, X, C, V, B, N, M. Below the keyboard, there are three input fields: "WORD:" with the value "MARTES", "LANGUAGE:" with the value "PORTUGUES", and "RESULT:" with the value "TERCA-FEIRA". At the bottom, there are two buttons: a light blue "TRANSLATE" button and a grey "CLEAN" button.