

DEFENSA HITO 4

JUAN ELIAN ALVAREZ VALLEJOS – SIS
PROGRAMACION III

Caso de uso: **DICCIONARIO (INGLES - ESPANOL - PORTUGUES)**

- Se tiene como objetivo generar una aplicacion de escritorio, en donde esta APP tendra la capacidad de traducir palabras en ESPAÑOL al idioma INGLES o PORTUGUES.
- En la primera fase (SPRINT 1) se implementara una prueba de concepto, es decir solo debera de mostrar traducciones de los dias de la semana.
- EJEM: Si el usuario escribe LUNES. (La aplicacion debere de mostrar su traduccion 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.

Diseño de la base de datos

- Se basara en el siguiente modelo.

Parte DESIGN:

DB

dictionary				
id	english	portugues	word	
111	MONDAY	SEGUNDA-FEIRA	LUNES	
112	TUESDAY	TERCA-FEIRA	MARTES	
113	WEDNESDAY	QUARTA-FEIRA	MIERCOLES	
114	THURSDAY	QUINTA-FEIRA	JUEVES	
115	FRIDAY	SEXTA-FEIRA	VIERNES	
116	SATURDAY	SABADO	SABADO	
117	SUNDAY	DOMINGO	DOMINGO	

PASO 1

- Debes crear tu Spring Application desde la pagina Spring.io.
- Colocar el nombre del Proyecto y sus dependencias – “Spring Web” “Spring Data JPA” “PostgreSQL Driver” y después presionar GENERATE.
- Como se muestra en la siguiente diapositiva.

Paso 1

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.Hito4

Artifact demo

Name demo

Description Demo project for Spring Boot

Package name com.Hito4.demo

Dependencies

ADD DEPENDENCIES... CTRL + B

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.

DEPENDENCIAS

GENERATE CTRL + G

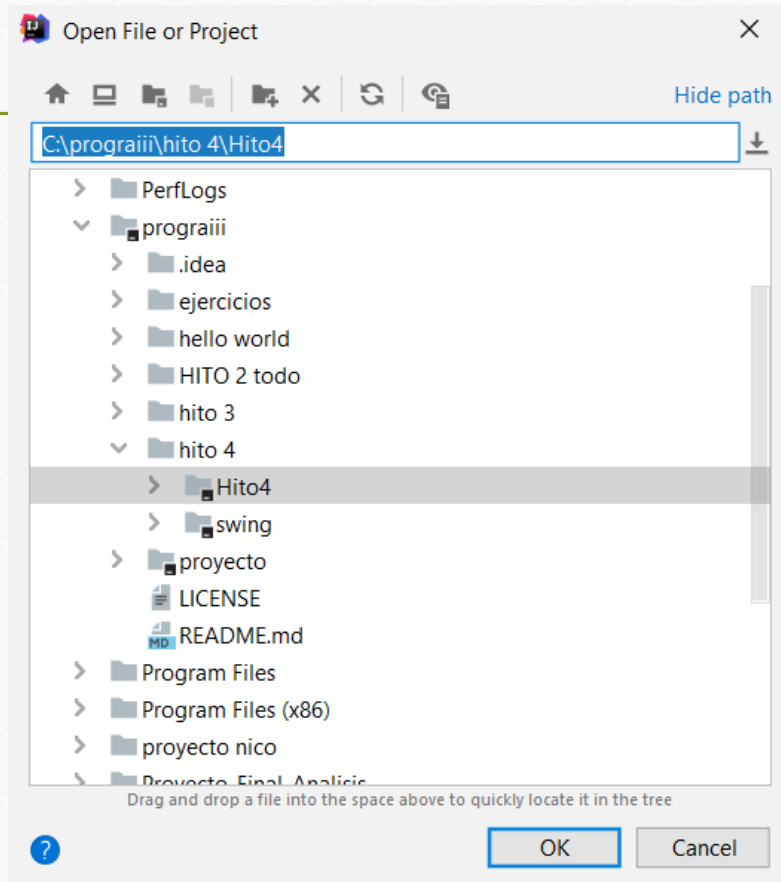
EXPLORE CTRL + SPACE

SHARE...

PASO 2

- Desde en IntelliJ IDEA se abre la carpeta que se descargo de la pagina Spring.io.
- Como en la siguiente diapositiva

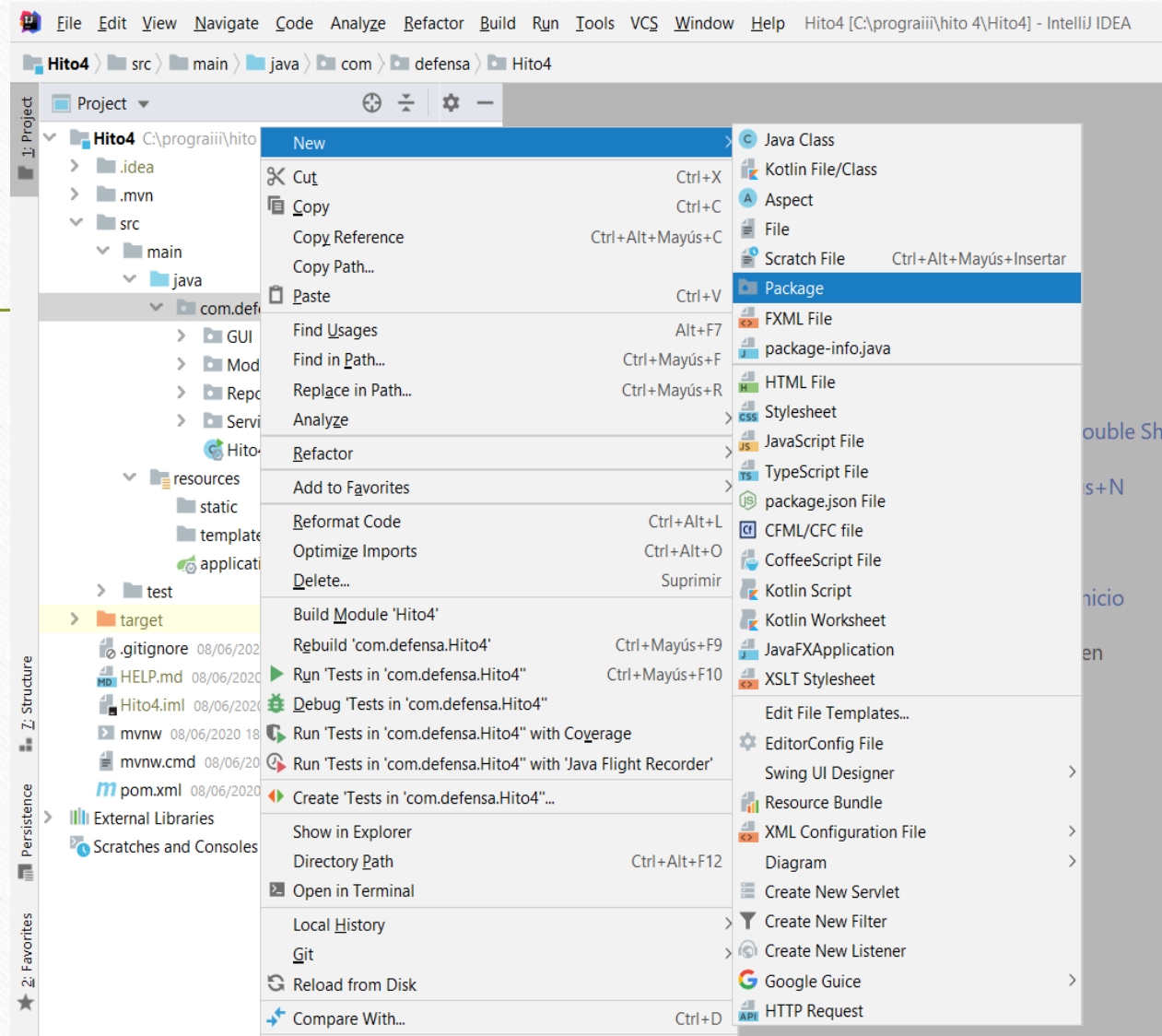
PASO 2



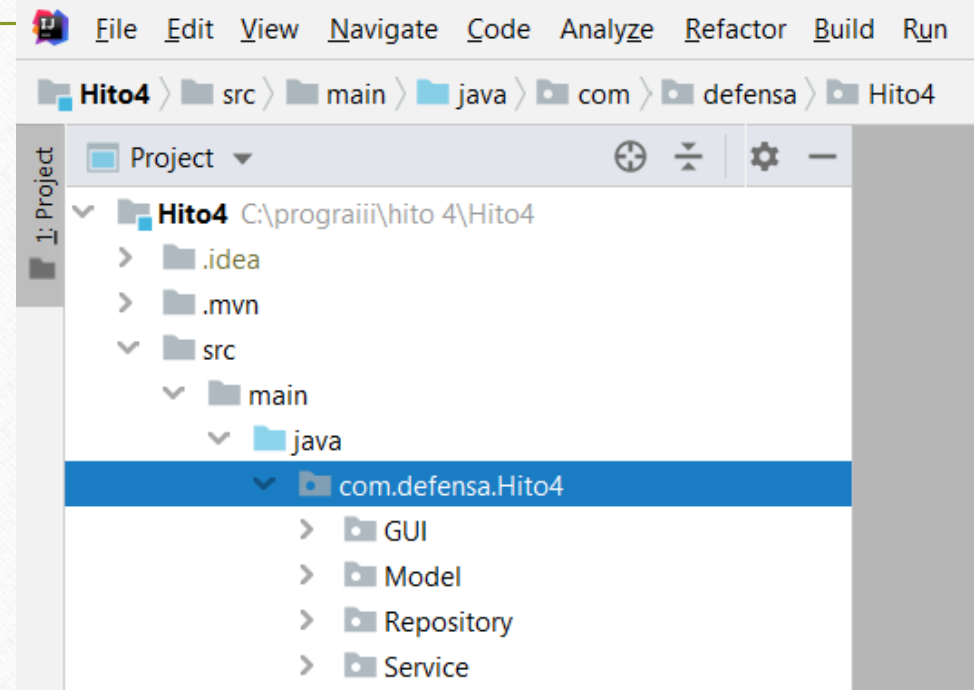
PASO 3

- Se debe crear todos los PACKAGES necesarios. Dando clic derecho en la carpeta que empiece su nombre por “com.”, después new y después en package.
- Como en las siguientes imagenes

PASO 3

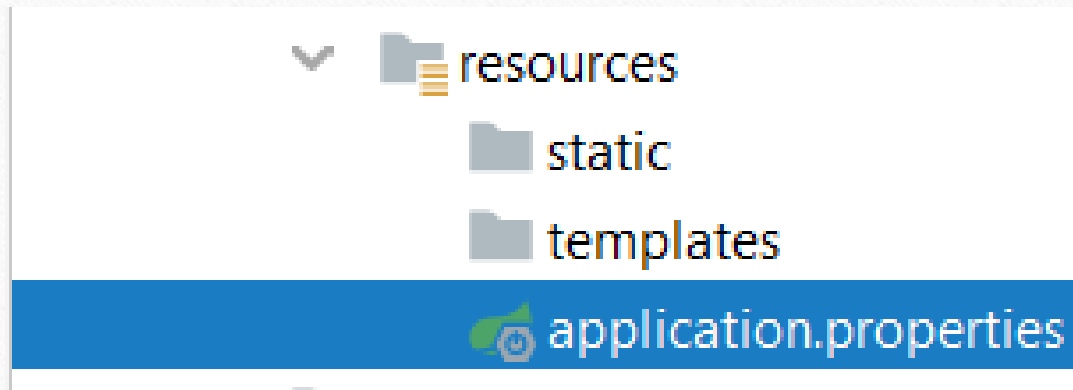


PASO 3



PASO 4

- Se debe conectar el IntelliJ IDEA con Datagrip.
- Para eso entramos en la carpeta “resources” y después “application.properties” y generar el siguiente código.



PASO 4

```
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/d3bhe5r04rr91p
spring.datasource.username=nnqpuwzgigklw
spring.datasource.password=4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

```
server.port = 8083
```

La url, el username y el password
Se debe colocar según las credenciales que te brinda Heruko como en la siguiente imagen.



PASO 5 - HERUKO


- Primero debes crearte una cuenta en heruko ingresando a heruko.com después creas un proyecto con el nombre que quieras, después entras a la opción Resources y a Add-ons para luego añadir un elemento llamado

Add-ons


Find more add-ons

Q Quickly add add-ons from Elements

 Heroku Postgres 

Attached as DATABASE 

Hobby Dev

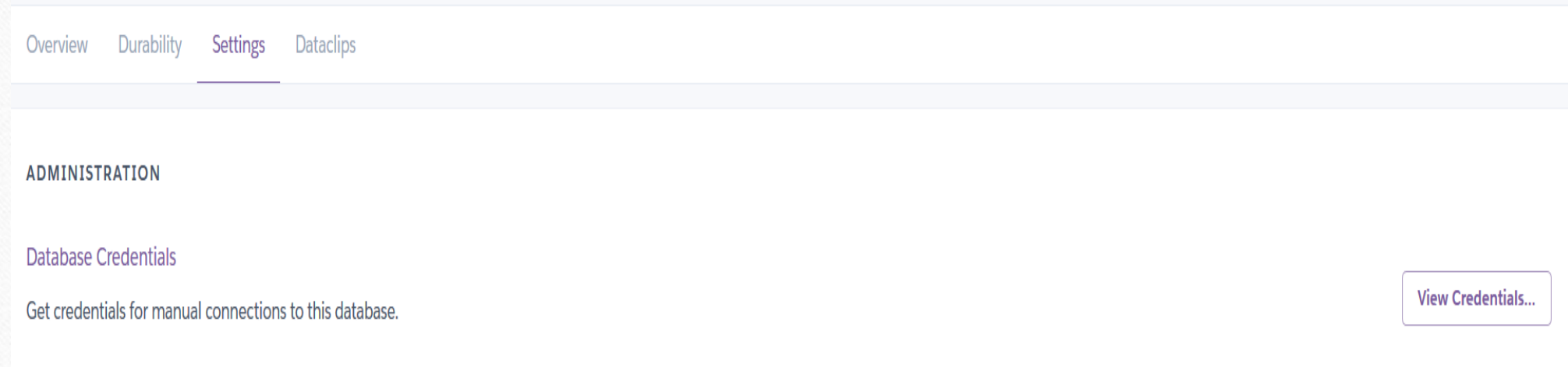
Free 

Estimated Monthly Cost

\$0.00

PASO 5 - HERUKO

- Después de darle clic te enviara a otra ventana. Y ahí vas a Settings y das clic a tus view credentials, donde veras tus credenciales.



PASO 5 – HERUKO - credenciales

- Esos datos que nos brinda HERUKO serán de gran utilidad ya que esos credenciales se colocara a Datagrip
- Y en IntelliJ IDEA.

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Cancel

Please note that **these credentials are not permanent**.

Heroku rotates credentials periodically and updates applications where this database is attached.

Host	ec2-52-201-55-4.compute-1.amazonaws.com
Database	d3bhe5r04rr9lp
User	nnqpuzgigklw
Port	5432
Password	4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6
URI	postgres://nnqpuzgigklw:4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6@ec2-52-201-55-4.compute-1.amazonaws.com:5432/d3bhe5r04rr9lp
Heroku CLI	heroku pg:psql postgresql-contoured-94333 --app prograiii2020elian

Ejercicio 1

Pregunta 1

US 1: Crear el modelo JPA-POSTGRESQL para el siguiente escenario.





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

	id	english	portugues
1	111	MONDAY	SEGUNDA-FEIRA
2	112	TUESDAY	TERÇA-FEIRA
3	113	WEDNESDAY	QUARTA-FEIRA
4	114	THURSDAY	QUINTA-FEIRA
5	115	FRIDAY	SEXTA-FEIRA
6	116	SATURDAY	SABADO
7	117	SUNDAY	DOMINGO

EJERCICIO 1

- Primero se crea la base de datos desde el IntelliJ Idea mediante código en java y con una conexión con heruko que después se conectara a Datagrip con las credenciales anteriormente mencionadas.
- HERUKO “Es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación y donde te crea una base de datos vía WEB y hace la conexión con Datagrip”.
- Datagrip – “Es una IDE multiplataforma para trabajar con SQL y base de datos.”

Ejercicio 1 – La base de datos desde Datagrip

	 id	 english	 portugues	 word
1	57	MONDAY	SEGUNDA-FEIRA	LUNES
2	58	TUESDAY	TERCA-FEIRA	MARTES
3	59	WEDNESDAY	QUARTA-FEIRA	MIERCOLES
4	60	THURSDAY	QUINTA-FEIRA	JUEVES
5	61	FRIDAY	SEXTA-FEIRA	VIERNES
6	62	SATURDAY	SABADO	SABADO
7	63	SUNDAY	DOMINGO	DOMINGO

Ejercicio 1 – el DDL de la base de datos

```
-- auto-generated definition
create table dictionary
(
    id            integer      not null
        constraint dictionary_pkey
            primary key,
    english       varchar(200) not null,
    portugues     varchar(200) not null,
    word          varchar(200) not null
);

alter table dictionary
    owner to nnqpuwzgigklw;
```

DDL
(Data Definition Language)

Ejercicio 1

- Como se menciono antes la base de datos se crea desde el código que generamos en IntelliJ IDEA – “en la clase Modelo” nos sirve para la tabla y después tendremos que rellenar de datos y lo haremos con el código de las “clases y sus interfaces”. Después de hacer eso simplemente debe hacer correr el programa.

Ejercicio 1

- El código del Modelo de las Letras y Los días de la semana esta dividido en 2 partes cada uno para que entre en la presentación pero al final todas las partes son uno solo.

Parte 1 modelo de las letras

```
package com.defensa.Hito4.Model;

import javax.persistence.*;

@Entity
@Table(name = "Alphabet2")
public class AlphabetModel2 {
    @javax.persistence.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;
```


Parte 2

```
public AlphabetModel2() {  
    }  
  
    public AlphabetModel2(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;  
    }  
}
```

Parte 1 modelo de los días

```
package com.defensa.Hito4.Model;

import javax.persistence.*;

@Entity
@Table(name = "dictionary")
public class DictionaryModel {
    @javax.persistence.Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer Id;

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

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

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

Parte 2

```
public DictionaryModel() {  
    }  
    public DictionaryModel(String english, String portugues, String word) {  
        this.english = english;  
        this.portugues = portugues;  
        this.word = word;  
    }  
    public DictionaryModel(String english, String first) {  
    }  
    public Integer getId() {  
        return Id;  
    }  
    public void setId(Integer id) {  
        Id = id;  
    }  
    public String getEnglish() {  
        return english;  
    }  
    public void setEnglish(String english) {  
        this.english = english;  
    }  
    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;  
    }  
}
```


Código de la clase “Service” de las letras

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

import com.defensa.Hito4.Model.AlphabetModel2;
import com.defensa.Hito4.Repository.AlphabetRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@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 AlphabetModel2(Q_P, "first"));
            alphabetRepository.save(new AlphabetModel2(A_L, "second"));
            alphabetRepository.save(new AlphabetModel2(Z_M, "three"));
        }
    }

    @Override
    public List<AlphabetModel2> getAllLettersFirst() {
        return alphabetRepository.getFirstRow();
    }

    @Override
    public List<AlphabetModel2> getAllLettersSecond() {
        return alphabetRepository.getSecondRow();
    }

    @Override
    public List<AlphabetModel2> getAllLettersThree() {
        return alphabetRepository.getThreeRow();
    }
}
```

Código de la interface “Service” de las letras

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

import com.defensa.Hito4.Model.AlphabetModel2;

import java.util.List;

public interface AlphabetServiceInterface {
    public void saveData();
    public List<AlphabetModel2> getAllLettersFirst();
    public List<AlphabetModel2> getAllLettersSecond();
    public List<AlphabetModel2> getAllLettersThree();
}
```

Código de la clase “Service” de los días

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

import com.defensa.Hito4.Model.AlphabetModel2;
import com.defensa.Hito4.Model.DictionaryModel;
import com.defensa.Hito4.Repository.AlphabetRepository;
import com.defensa.Hito4.Repository.DictionaryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class DictionaryService implements DictionaryServiceInterface{
    @Autowired
    private DictionaryRepository dictionaryRepository;

    @Override
    public void saveData() {
        if (dictionaryRepository.count() == 0) {

            dictionaryRepository.save(new DictionaryModel("MONDAY", "SEGUNDA-FEIRA", "LUNES"));
            dictionaryRepository.save(new DictionaryModel("TUESDAY", "TERCA-FEIRA", "MARTES"));
            dictionaryRepository.save(new DictionaryModel("WEDNESDAY", "QUARTA-FEIRA", "MIERCOLES"));
            dictionaryRepository.save(new DictionaryModel("THURSDAY", "QUINTA-FEIRA", "JUEVES"));
            dictionaryRepository.save(new DictionaryModel("FRIDAY", "SEXTA-FEIRA", "VIERNES"));
            dictionaryRepository.save(new DictionaryModel("SATURDAY", "SABADO", "SABADO"));
            dictionaryRepository.save(new DictionaryModel("SUNDAY", "DOMINGO", "DOMINGO"));

        }
    }
}
```


Código de la interface “Service” de los días

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

import com.defensa.Hito4.Model.AlphabetModel2;
import com.defensa.Hito4.Model.DictionaryModel;

import java.util.List;

public interface DictionaryServiceInterface {
    public void saveData();
    //    public List<DictionaryModel> getAllLettersFirst();
    //    public List<DictionaryModel> getAllLettersSecond();
    //    public List<DictionaryModel> getAllLettersThree();
}
```

Código del main

```
package com.defensa.Hito4;

import com.defensa.Hito4.Service.AlphabetService;
import com.defensa.Hito4.Service.DictionaryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;

import javax.annotation.PostConstruct;

@SpringBootApplication
public class Hito4Application {
    @Autowired
    private AlphabetService AlphabetService;
    @Autowired
    private DictionaryService DictionaryService;

    public static void main(String[] args) {
        //SpringApplication.run(SwingApplication.class, args);
        SpringApplicationBuilder springApp = new SpringApplicationBuilder(Hito4Application.class);
        springApp.headless(false);
        springApp.run(args);
    }

    @PostConstruct
    public void insertDataToDataBase(){

        AlphabetService.saveData();
        DictionaryService.saveData();
    }
}
```

Ejercicio 2

Pregunta 2

US 2: Implementar el diseño SWING para la siguiente ventana.

The image shows a graphical user interface (GUI) design for a translator application. The interface is contained within a light grey rectangular frame. At the top, there is a keyboard layout with three rows of letter buttons. The first row contains buttons for Q, W, E, R, T, Y, U, I, O, and P. The second row contains buttons for A, S, D, F, G, H, J, K, and L. The third row contains buttons for Z, X, C, V, B, N, and M. All buttons are light blue with dark blue text. Below the keyboard, there are three input fields. The first is labeled 'WORD:', the second is labeled 'LANGUAGE:', and the third is labeled 'RESULT:'. Each label is followed by a white rectangular input box. At the bottom of the frame, there are two buttons. The first button is labeled 'TRANSLATE' and is light blue. The second button is labeled 'CLEAN' and is dark grey.

Nota. Solo diseño, no funcionalidad.

Ejercicio 2

- Para poder crear el diseño de la ventana que mostrara el traductor se debe crear las siguientes clase y los paneles.
- MainFrame
- PanelPrincipal
- Panel de las cajas de texto y los labels
- Panel de los 2 buttons
- La clase UTIL para el diseño de las letras

Código del MainFrame parte 1

```
package com.defensa.Hito4.GUI.Frame;

import com.defensa.Hito4.GUI.Panels.EmptyPanel;
import com.defensa.Hito4.GUI.Panels.Panel2buttons;
import com.defensa.Hito4.GUI.Panels.PanelPrincipal;
import com.defensa.Hito4.GUI.Panels.PanelcajasLabels;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import javax.annotation.PostConstruct;
import javax.swing.*;
import java.awt.*;

@Component
public class MainFrame extends JFrame {
    // @Autowired
    // private EmptyPanel emptyPanel;
    @Autowired
    private PanelPrincipal panelPrincipal;
    @Autowired
    private PanelcajasLabels panelcajasLabels;
    @Autowired
    private Panel2buttons panel2buttons;

    public MainFrame() {
        this.setTitle("PROGRA III 2020");
        this.setBounds(300, 200, 800, 500);
        this.setBackground(Color.blue);
        this.setLayout(new GridLayout(1, 0));
        //this.setResizable(false);
    }
}
```

Código del MainFrame parte 2

```
@PostConstruct
public void createPanelsMainFrame() {
    JPanel container = new JPanel();
    container.setLayout(new FlowLayout());

    addPanels(container);

    this.add(container);
    this.setVisible(true);
}

public void addPanels(JPanel container) {
    container.add(panelPrincipal);
    container.add(panelcajasLabels);
    container.add(panel2buttons);
    //container.add(emptyPanel);
}
}
```


Código del panel principal parte 1

```
package com.defensa.Hito4.GUI.Panels;

import com.defensa.Hito4.GUI.Listener.ButtonListener;
import com.defensa.Hito4.GUI.util.Util;
import com.defensa.Hito4.Model.AlphabetModel2;
import com.defensa.Hito4.Service.AlphabetService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.List;

@Component
public class PanelPrincipal extends JPanel {

    @Autowired
    private AlphabetService alphabetService;

    public PanelPrincipal() {
        System.setProperty("butBackColor", "#C1ECF1");
        System.setProperty("textColor", "#0B0BF6");
        this.setPreferredSize(new Dimension(600, 190));
        this.setBackground(Color.bLue);
        this.setLayout(new GridLayout(3, 0));
    }
}
```

Código del panel principal parte 2

```
@PostConstruct
public void createButtonsLetters() {
    java.util.List<AlphabetModel2> firstRow = alphabetService.getAllLettersFirst();
    String[] titleAlphabet = firstRow.get(0).getLetter().split(",");
    JPanel panelQ_P = this.createPanelButton(titleAlphabet);
    this.add(panelQ_P);

    java.util.List<AlphabetModel2> secondRow = alphabetService.getAllLettersSecond();
    String[] titleAlphabet1 = secondRow.get(0).getLetter().split(",");
    JPanel panelA_L = this.createPanelButton(titleAlphabet1);
    this.add(panelA_L);

    List<AlphabetModel2> threeRow = alphabetService.getAllLettersThree();
    String[] titleAlphabet2 = threeRow.get(0).getLetter().split(",");
    JPanel panelZ_M = this.createPanelButton(titleAlphabet2);
    this.add(panelZ_M);
}

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(55, 40));
        button.addActionListener(listener);
        button.setBackground(Color.getColor("butBackColor"));
        button.setForeground(Color.getColor("textColor"));
        button.setBorder(BorderFactory.createEmptyBorder());
        button.setFont(Util.FONT_TEXT);
        mainPanel.add(button);
    }
    return mainPanel;
}
```

Código del Panel cajasLabel parte 1

```
package com.defensa.Hito4.GUI.Panels;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.swing.*;
import java.awt.*;

@Component
public class PanelcajasLabels extends JPanel {

    public PanelcajasLabels(){
        this.setPreferredSize(new Dimension(900, 100));
        //this.setBounds(50, 500, 900, 100);
        this.setLayout(new GridLayout(1, 0));
    }
}
```


@PostConstruct

```
public void createButtonsLabel() {  
    JPanel caja1 = this.createTextBox();  
    this.add(caja1);  
}  
  
public JPanel createTextBox()  
{  
    JPanel mainPanel = new JPanel();  
    mainPanel.setLayout(new FlowLayout());  
  
    JLabel label = new JLabel("WORD:");  
    label.setPreferredSize(new Dimension(50, 40));  
  
    JTextField txtIVA = new JTextField();  
    txtIVA.setPreferredSize(new Dimension(100, 30));  
  
    JLabel label2 = new JLabel("LANGUAGE:");  
    label2.setPreferredSize(new Dimension(75, 40));  
  
    JTextField txtIVA2 = new JTextField();  
    txtIVA2.setPreferredSize(new Dimension(100, 30));  
  
    JLabel label3 = new JLabel("RESULT:");  
    label3.setPreferredSize(new Dimension(60, 40));  
  
    JTextField txtIVA3 = new JTextField();  
    txtIVA3.setPreferredSize(new Dimension(100, 30));  
    mainPanel.add(label);  
    mainPanel.add(txtIVA);  
    mainPanel.add(label2);  
    mainPanel.add(txtIVA2);  
    mainPanel.add(label3);  
    mainPanel.add(txtIVA3);  
  
    return mainPanel;  
}  
}
```

Código del Panel cajasLabel parte 2

Código del Panel de los 2 buttons

parte 1

```
package com.defensa.Hito4.GUI.Panels;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;

@Component
public class Panel2buttons extends JPanel {

    public Panel2buttons(){
        System.setProperty("butBackColor", "#C1ECF1");
        System.setProperty("textColor", "#0B0BF6");

        this.setPreferredSize(new Dimension(400, 190));
        this.setLayout(new GridLayout(1, 0));
    }
}
```

Código del Panel de los 2 buttons parte 2

```
@PostConstruct
public void create2Buttons() {
    JPanel caja2 = this.create2buttons();
    this.add(caja2);
}
public JPanel create2buttons()
{
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new FlowLayout());

    JButton button = new JButton("TRANSLATE");
    button.setPreferredSize(new Dimension(150, 50));
    button.setBackground(Color.getColor("butBackColor"));
    button.setForeground(Color.getColor("textColor"));

    JButton button2 = new JButton("CLEAN");
    button2.setPreferredSize(new Dimension(100, 50));
    button2.setBackground(Color.gray);
    button2.setForeground(Color.getColor("textColor"));

    mainPanel.add(button);
    mainPanel.add(button2);
    return mainPanel;
}
}
```


Codigo de la clase UTIL

```
package com.defensa.Hito4.GUI.util;

import org.springframework.stereotype.Component;

import javax.swing.*;
import java.awt.*;
import java.net.URL;

@Component
public class Util {

    public static final Font FONT_TEXT = new Font("Arial", Font.BOLD, 28);
    public Util(){}
}
```

Ejercicio 2

- Una vez echo el código solo faltaría que apreten RUN y les mostrara el diseño visto en la siguiente diapositiva.

Ejercicio 2 resultado

JB PROGRA III 2020

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

Ejercicio 3

Pregunta 3

US 3: Implementar la funcionalidad de la APP.

WORD: LUNES LANGUAGE: INGLES RESULT: MONDAY

TRANSLATE CLEAN

Sugerencia puede utilizar algo similar en una query custom para poder obtener los registros de la DB. (Solo si considera necesario)

```
@Query(value = "SELECT * FROM dictionary WHERE word = :wordSelected", native = true)  
public DictionaryModel getWordTranslate(@Param("wordSelected") String wordSelected)
```

Ejercicio 3

- Para hacer funcionar el traductor debemos utilizar un *Acciónlistener* para que los botones funcionen y colocar algunas funciones en sus interfaz y en su repositorio utilizar una consulta que hará que obtengamos la palabra traducida.
- Utilizaremos:
- Repository
- Interfaz
- Service
- Panel

Código del Repositorio de los días

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

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

import java.util.List;

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


Código de la interfaz de los días

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

import com.defensa.Hito4.Model.AlphabetModel2;
import com.defensa.Hito4.Model.DictionaryModel;

import java.util.List;

public interface DictionaryServiceInterface {
    public void saveData();
    String traducir(String t, String l);
}
```

Esta es la función que aumentamos para poder traducir



Código del Service de los días

```
@Override
public String traducir(String t, String l) {
    DictionaryModel dictionaryModel = dictionaryRepository.getWordTranslate(t);
    String ingles = dictionaryModel.getEnglish();
    String ln_i = "INGLES";
    String word = dictionaryModel.getWord();
    String ln_w = "ESPAÑOL";
    String portugues = dictionaryModel.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;
}
```

Solo se aumenta esta función
Donde comparamos que
idioma será

Código del Panel

- Anteriormente creamos los paneles cajasLabel y 2buttons pero para poder dar solución los unimos ambos paneles y de ahí el nuevo código del panel.

Código del Panel parte 1

```
package com.defensa.Hito4.GUI.Panels;

import com.defensa.Hito4.Service.DictionaryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

@Component
public class Panel2buttons extends JPanel{

    @Autowired
    // private PanelcajasLabels panelcajasLabels;
    private DictionaryService dictionaryService;

    public Panel2buttons(){
        System.setProperty("butBackColor", "#C1ECF1");
        System.setProperty("textColor", "#0B0BF6");

        this.setPreferredSize(new Dimension(600, 1000));
        this.setLayout(new GridLayout(2, 0));
    }

    @PostConstruct
    public void create2Buttons() {
        JPanel caja2 = this.create2buttons();
        this.add(caja2);
        //*****
        /* JPanel caja1 = this.create2buttons();
        this.add(caja1);*/
    }
}
```

Código del panel parte 2

```
public JPanel create2buttons()
{
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new FlowLayout());
    JLabel label = new JLabel("WORD:");
    label.setPreferredSize(new Dimension(50, 40));

    JTextField txtIVA = new JTextField();
    txtIVA.setPreferredSize(new Dimension(100, 30));

    JLabel label2 = new JLabel("LANGUAGE:");
    label2.setPreferredSize(new Dimension(75, 40));

    JTextField txtIVA2 = new JTextField();
    txtIVA2.setPreferredSize(new Dimension(100, 30));

    JLabel label3 = new JLabel("RESULT:");
    label3.setPreferredSize(new Dimension(60, 40));

    JTextField txtIVA3 = new JTextField();
    txtIVA3.setPreferredSize(new Dimension(100, 30));

    JButton button = new JButton("TRANSLATE");
    button.setPreferredSize(new Dimension(150, 50));
    button.setBackground(Color.getColor("butBackColor"));
    button.setForeground(Color.getColor("textColor"));

    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            JButton button = (JButton) actionEvent.getSource();
            String w = txtIVA.getText();
            String l = txtIVA2.getText();
            String t = dictionaryService.traducir(w,l);
            txtIVA3.setText(t);
        }
    });
}
```

Código del panel parte 3

```
JButton button2 = new JButton("CLEAN");
button2.setPreferredSize(new Dimension(100, 50));
button2.setBackground(Color.gray);
button2.setForeground(Color.getColor("textColor"));

button2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        JButton button2 = (JButton) actionEvent.getSource();
        String clear = "";
        txtIVA.setText(clear);
        txtIVA2.setText(clear);
        txtIVA3.setText(clear);
    }
});

mainPanel.add(label);
mainPanel.add(txtIVA);
mainPanel.add(label2);
mainPanel.add(txtIVA2);
mainPanel.add(label3);
mainPanel.add(txtIVA3);
mainPanel.add(button);
mainPanel.add(button2);
return mainPanel;
}
```

Este código entra en el código de la anterior diapositiva justo debajo.

Código del Mainframe Modificado

- Modificamos el mainframe porque unimos los paneles.
- Se mostrara en 2 partes

Codigo Parte 1 del Mainframe

```
package com.defensa.Hito4.GUI.Frame;

import com.defensa.Hito4.GUI.Panels.EmptyPanel;
import com.defensa.Hito4.GUI.Panels.Panel2buttons;
import com.defensa.Hito4.GUI.Panels.PanelPrincipal;
//import com.defensa.Hito4.GUI.Panels.PanelcajasLabels;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.swing.*.*;
import java.awt.*.*;

@Component
public class MainFrame extends JFrame {
    // @Autowired
    // private EmptyPanel emptyPanel;
    @Autowired
    private PanelPrincipal panelPrincipal;
    // @Autowired
    // private PanelcajasLabels panelcajasLabels;
    @Autowired
    private Panel2buttons panel2buttons;

    public MainFrame() {
        this.setTitle("PROGRA III 2020");
        this.setBounds(300, 200, 800, 500);
        this.setBackground(Color.blue);
        this.setLayout(new GridLayout(1, 0));
        //this.setResizable(false);
    }
}
```

Código Parte 2 del Mainframe

```
@PostConstruct
public void createPanelsMainFrame() {
    JPanel container = new JPanel();
    container.setLayout(new FlowLayout());

    addPanels(container);

    this.add(container);
    this.setVisible(true);
}

public void addPanels(JPanel container) {
    container.add(panelPrincipal);
    //container.add(panelCajasLabels);
    container.add(panel2buttons);
    //container.add(emptyPanel);
}
}
```


Resultado esperado

JB PROGRA III 2020

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:

GRACIAS !!!!