

Instructivo de aplicación Entrevista Prueba para fonYou Colombia S.A.S

Desarrollado Por Nicolas Vargas

Bogotá - Colombia

2022

1.1 Resumen

Mucho gusto, me llamo Nicolas Vargas y a continuación les explicare como elabore la prueba técnica y todas las tecnologías que utilice para darle forma

1.2 Tecnologías Utilizadas

- Java 11
- Spring Framework
- Spring Tool Suite 4
- Bootstrap 5
- Thymeleaf
- Maven
- Mysql (XAMPP)
- PostMan
- HTML5

1.3 Presentación Capa De Modelo

El proyecto cuenta con una interfaz de inicio para mayor comodidad, además de estar mapeado con el modelo de estudiante que vamos a pasar hacia un método post

EXAMEN

Acceder

Una vez ingresamos podremos diligenciar los datos del estudiante que va a presentar el examen como:

-Nombre

-Edad

-Ciudad

-Zona Horaria

Tal como se especifico en los requerimientos de la prueba, el botón continuar ejecuta un método post que va a guardar la información del estudiante en la base de datos que nombre fonyoubd con dos tablas, en la tabla estudiantes es donde se va a almacenar esta información

EXAMEN

Diligencie los siguientes datos para continuar

Nombre

Edad

Ciudad

Zona Horaria

Continuar

©FontYou Prueba Técnica - Nicolas vargas

En un punto de requerimientos se nos pide recolectar la zona horaria del usuario, para este ejemplo puse 5 Zonas distintas, de las cuales Bogotá esta seleccionada por defecto, sin embargo con la clase ZonedDateTime recopiló las zonas horarias

Zona Horaria

(GMT-05:00) Bogota, Lima, Quito, Rio Branco

(GMT-06:00) Central Time (US - Canada)

(GMT-06:00) Guadalajara, Mexico City, Monterrey

(GMT-05:00) Bogota, Lima, Quito, Rio Branco

(GMT-03:00) America, Argentina, Cordoba

(GMT-08:00) Hongkong

Cuando diligenciamos los datos y pasamos a la siguiente pagina es momento de presentar el examen

Escogí crear un examen con 4 preguntas de java, cada respuesta se almacenara en la base de datos y cada que presentamos el examen van a cambiar de orden, más adelante explicare como lo realice, hice las respectivas validaciones para que solo se pudiera escoger un radio button y darle el valor respectivo que se haya escogido para hacer el post en la base de datos, inserte un pequeño botón utilizando javascript con un pequeño mensaje :D.

EXAMEN

Preguntas

¿ Cual es el lenguaje de programación más famoso y por que es java ?

- ☐ Python
☐ c
☐ Java
☐ Javascript

¿ Quien es el creador de java ?

- ☐ Bjarne Stroustrup
☐ James Gosling
☐ Yukihiko Matsumoto
☐ Guido van Rossum

¿ En que año fue desarrollado java ?

- ☐ 1992
☐ 1999
☐ 2001
☐ 1995

¿Qué código de los siguientes tiene que ver con la herencia?

- ☐ public class Componente belong to Producto
☐ public class Componente implements Producto
☐ public class Componente extends Producto
☐ public class Componente inherit Producto

Enviar

Click Aqui :D

Una vez enviamos la información nos va a llevar a la pagina donde nos muestra la calificación que sacamos mostrando el nombre, las notas, la calificación final y nuestra zona horaria

EXAMEN

Nombre	Nota1	Nota2	Nota3	Nota4	Calificacion	Zona Horaria
Oscar	25	0	25	25	75	2022-11-04T16:39:29.307613-05:00[America/Bogota]

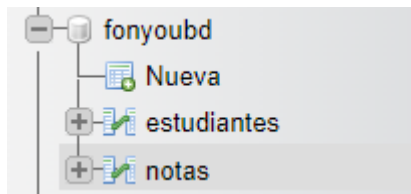
Por lo tanto, en la capa de vista contaremos con 4 paginas con su respectiva funcionalidad cada una

CAPA DE DATOS

La capa de datos la elabore utilizando MySQL con phpMyAdmin ejecutando el servidor en XAMPP

La base de datos se llama fonyoubd

Y contamos con dos tablas: Estudiantes y Notas



En la tabla de estudiantes las adjuntare con 3 registros que realice para pruebas y verificar una correcta funcionalidad:

Contamos con los campos:

-id_estudiante

-Nombre

-Edad

-Ciudad

-Zona_horaria

<div><div><div>←</div><div>T</div><div>→</div></div><div></div></div>				id_estudiante	nombre	edad	ciudad	zona_horaria
<div><div><div></div></div><div><div><div></div></div></div><div><div>Editar</div><div>Copiar</div><div>Borrar</div></div></div>	1	nicolas	23	Bogota	America/Bogota			
<div><div><div></div></div><div><div><div></div></div></div><div><div>Editar</div><div>Copiar</div><div>Borrar</div></div></div>	7	sofia	25	Bogota	America/Bogota			
<div><div><div></div></div><div><div><div></div></div></div><div><div>Editar</div><div>Copiar</div><div>Borrar</div></div></div>	8	Oscar	32	Bogota	America/Bogota			

En la tabla notas se adjuntarán las respuestas del usuario que luego en el controlador serán validadas para dar un puntaje y sacar una calificación










-id_notas

-nota1

-nota2

-nota3

-nota4

	id_notas	nota1	nota2	nota3	nota4
<input type="checkbox"/>  Editar  Copiar  Borrar	1	Java	Yukihiro Matsumoto	1995	public class Componente extends Producto
<input type="checkbox"/>  Editar  Copiar  Borrar	7	Javascript	James Gosling	1995	public class Componente extends Producto
<input type="checkbox"/>  Editar  Copiar  Borrar	8	Java	Yukihiro Matsumoto	1995	public class Componente extends Producto

CAPA DE NEGOCIO

Lo primero que configure fue el pom, para integrar las tecnologías necesarias y especificar las versiones y dependencias que voy a utilizar

```
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.5</version>
  <relativePath /> <!-- lookup parent from repository -->
</parent>
<groupId>com.entrevista</groupId>
<artifactId>EntrevistaTecnica</artifactId>
<version>0.0.1</version>
<name>PruebaTecnica</name>
<description>entrevista tecnica programador nicolas
vargas</description>
<properties>
  <java.version>11</java.version>
</properties>
```

Y aquí las dependencias que utilice.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web-services</artifactId>
```

```

</dependency>

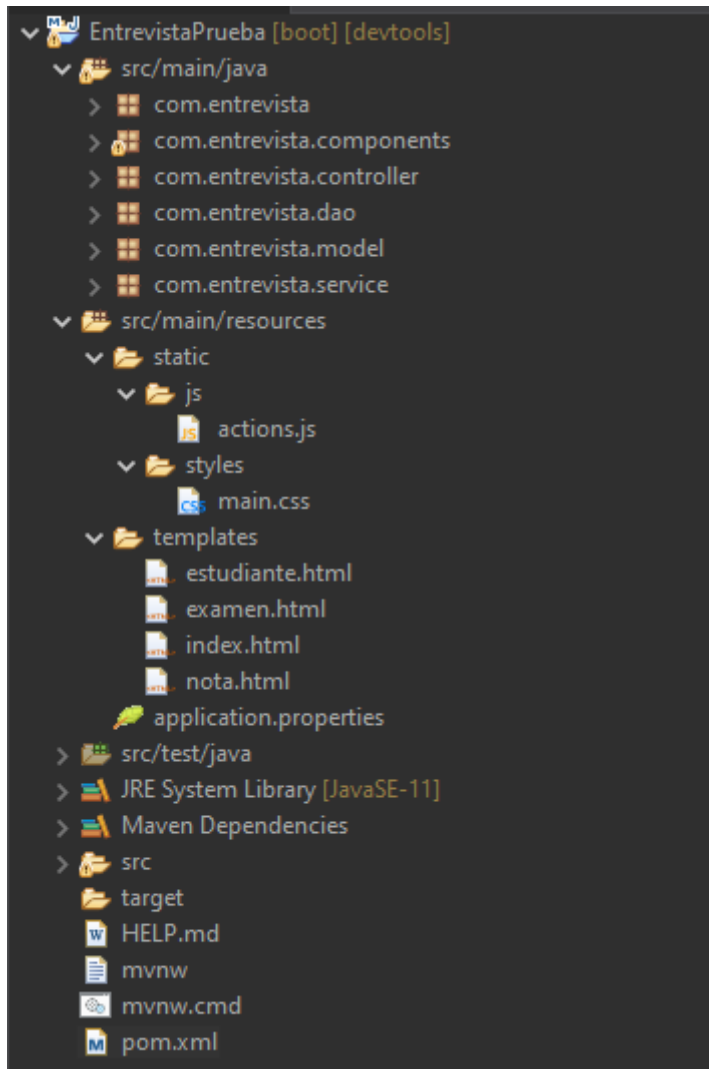
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>5.0.0</version>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
</dependency>
</dependencies>

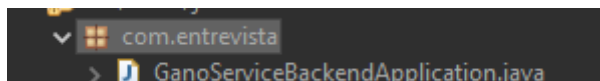
```

Como podrán observar utilice JPA y msql, también para la capa de vista utilice thymeleaf para tenerlo todo integrado en spring boot

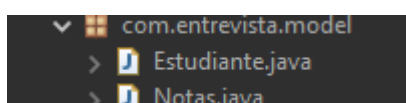
Los paquetes del proyecto se subdividen así para obtener una buena modularización del código:



En el paquete com.entrevista tenemos el contenedor principal de spring que arranca nuestro proyecto



En el paquete com.entrevista.model tenemos los modelos de entidad:



Les mostrare el modelo de entidad de persona:

Cuenta con sus respectivos set y gets, en otro posible caso hubiera podido utilizar lombok, pero para este caso preferí hacerlo manual

```
package com.entrevista.model;

import java.io.Serializable;

@Entity
@Table(name = "estudiantes")
public class Estudiante implements Serializable{

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name="id_estudiante")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String nombre;
    private int edad;
    private String ciudad;
    @Column(name="zona_horaria")
    private String zona;

    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
}
```

En el paquete com.entrevista.dao

Cree el data Access object para utilizar los métodos crud que nos brinda JPA, cree un dao para notas y otro para estudiantes

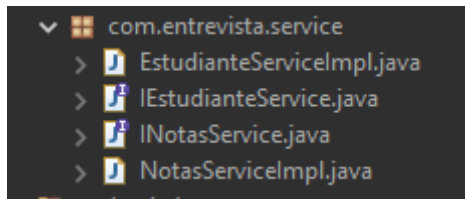
```
com.entrevista.dao
├── EstudianteDao.java
└── NotasDao.java
```

```
import org.springframework.data.repository.CrudRepository;

public interface EstudianteDao extends CrudRepository<Estudiante, Long>{

}
```

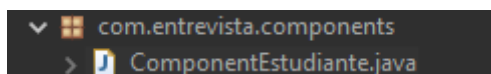
En el paquete com.entrevista.service cree toda la capa de negocio, utiliza una interfaz y luego la implementación de la interfaz, para mas tarde en el controlador conectarnos directamente a la interfaz



Les mostrare la implementación de estudiante Service, donde cree los métodos para leer el listado de estudiantes ingresados y guardar sus datos

```
2
3 @Service
4 public class EstudianteServiceImpl implements IEstudianteService{
5
6
7     @Autowired
8     private EstudianteDao estudianteDao;
9
10
11     @Override
12     @Transactional(readOnly = true)
13     public List<Estudiante> listarPersonas() {
14
15
16         return (List<Estudiante>) estudianteDao.findAll();
17     }
18
19     @Override
20     @Transactional
21     public void guardar(Estudiante estudiante) {
22         estudianteDao.save(estudiante);
23     }
24
25
26
27 }
28
```

En el paquete com.entrevista.components cree una clase java para ayudarme a obtener las preguntas del examen y la zona horaria



Gracias a esta clase pude crear y obtener las preguntas y la zona horaria

```

@Component
public class ComponentEstudiante {

    private String pregunta1 = "¿ Cual es el lenguaje de programación más famoso y por que es java ?",
        pregunta2 = "¿ Quien es el creador de java ?",
        pregunta3 = "¿ En que año fue desarrollado java ?",
        pregunta4 = "¿Qué código de los siguientes tiene que ver con la herencia?";

    public String getPregunta1() {
        return pregunta1;
    }

    public void setPregunta1(String pregunta1) {
        this.pregunta1 = pregunta1;
    }

    public String getPregunta2() {
        return pregunta2;
    }

    public void setPregunta2(String pregunta2) {
        this.pregunta2 = pregunta2;
    }

    public String getPregunta3() {
        return pregunta3;
    }

    public void setPregunta3(String pregunta3) {
        this.pregunta3 = pregunta3;
    }

    public String getPregunta4() {
        return pregunta4;
    }

    public void setPregunta4(String pregunta4) {
        this.pregunta4 = pregunta4;
    }

    public Collection<String> preguntaUnoArray() {
        var opciones1 = Arrays.asList("Java", "Python", "c", "Javascript" );
        Collections.shuffle(opciones1);
    }
}

```

```

    public Collection<String> preguntaUnoArray() {
        var opciones1 = Arrays.asList("Java", "Python", "c", "Javascript" );
        Collections.shuffle(opciones1);
        return opciones1;
    }

    public Collection<String> preguntaDosArray() {
        var opciones2 = Arrays.asList("Guido van Rossum", "Yukihiro Matsumoto", "James Gosling", "Bjarne Stroustrup" );
        Collections.shuffle(opciones2);
        return opciones2;
    }

    public Collection<String> preguntaTresArray() {
        var opciones3 = Arrays.asList("1995", "1992", "2001", "1999" );
        Collections.shuffle(opciones3);
        return opciones3;
    }

    public Collection<String> preguntaCuatroArray() {
        var opciones4 = Arrays.asList("public class Componente inherit Producto", "public class Componente extends Producto",
            "public class Componente belong to Producto", "public class Componente implements Producto" );
        Collections.shuffle(opciones4);
        return opciones4;
    }

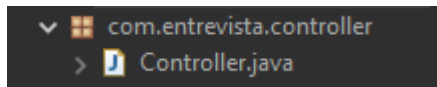
    public ZonedDateTime obtenerZonaHoraria(ZoneId zone) {
        //obtener zonas horarias
        // Set<String> availableZoneIds = ZoneId.getAvailableZoneIds();
        // for (String zoneId : availableZoneIds) {
        //     System.out.println(zoneId);
        // }

        //zonas horarias que aplique
        //ZoneId zoneIdBg = ZoneId.of("America/Bogota");
        //ZoneId zoneIdMx = ZoneId.of("America/Mexico_City");
        //ZoneId zoneIdUs = ZoneId.of("US/Central");

        return ZonedDateTime.now(zone);
    }
}

```

En el paquete com.entrevista.controller



Es donde implemente todos los métodos necesarios para mapear la información, pasarla a la vista y almacenar la información como recuperarla de la base de datos, además de inyectar los servicios

El metodo thyme simplemente nos devuelve el titulo de la pagina que utilizaremos en todos los html

Luego tenemos el metodo get de /agregar donde pasamos el objeto estudiante que se va asociar al formulario y luego el metodo post /guardar que se encarara de guardar a nuestro estudiante en la base de datos haciendo una validación de la información y redigiriendo hacia la siguiente pagina

```
37 import java.time.ZoneId;
38
39 @org.springframework.stereotype.Controller
40 public class Controller {
41
42     @Autowired
43     private IEstudianteService estudianteService;
44
45     @Autowired
46     private INotasService notasService;
47
48     @GetMapping("/")
49     public String thyme(Model model) {
50
51         String mensaje = "Examen";
52
53         model.addAttribute("mensaje", mensaje);
54
55         // devuelve la pagina que se va a desplegar en el navegador en este caso index
56         return "index";
57     }
58
59     @GetMapping("/agregar")
60     public String agregar(Estudiante estudiante) {
61         return "estudiante";
62     }
63
64     @PostMapping("/guardar")
65     public String guardar(@Validated Estudiante estudiante, Errors errores) {
66
67         if (errores.hasErrors()) {
68             // si tenemos errores retornamos al index
69             return "redirect:/";
70         }
71
72         estudianteService.guardar(estudiante);
73         return "redirect:/examen";
74     }
75
76     @GetMapping("/examen")
77     public String examen(Notas notas, Model model) {
78
79         ComponentEstudiante cEstudiante = new ComponentEstudiante();
80
81         var pregunta1 = cEstudiante.getPregunta1();
82         model.addAttribute("pregunta1", pregunta1);
83         var pregunta2 = cEstudiante.getPregunta2();
```

En el metodo get /examen vamos a crear objetos de las preguntas y vamos a pasarlos a la vista, en el componente almacene las preguntas en un array y luego con el metodo shuffle las puse en orden aleatorio para que cada vez que ingresemos el orden sea diferente, paramos el titulo de la pregunta y también paramos el array de preguntas

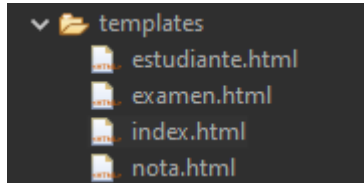
En el metodo guardarNotas es el que se encarga de almacenar las respuestas del estudiante en la base de datos

```
5
6
7 @GetMapping("/examen")
8 public String examen(Notas notas, Model model) {
9
10     ComponentEstudiante cEstudiante = new ComponentEstudiante();
11
12     var pregunta1 = cEstudiante.getPregunta1();
13     model.addAttribute("pregunta1", pregunta1);
14     var pregunta2 = cEstudiante.getPregunta2();
15     model.addAttribute("pregunta2", pregunta2);
16     var pregunta3 = cEstudiante.getPregunta3();
17     model.addAttribute("pregunta3", pregunta3);
18     var pregunta4 = cEstudiante.getPregunta4();
19     model.addAttribute("pregunta4", pregunta4);
20
21     var arrayPregunta1 = cEstudiante.preguntaUnoArray();
22     model.addAttribute("arrayPregunta1", arrayPregunta1);
23
24     var arrayPregunta2 = cEstudiante.preguntaDosArray();
25     model.addAttribute("arrayPregunta2", arrayPregunta2);
26
27     var arrayPregunta3 = cEstudiante.preguntaTresArray();
28     model.addAttribute("arrayPregunta3", arrayPregunta3);
29
30     var arrayPregunta4 = cEstudiante.preguntaCuatroArray();
31     model.addAttribute("arrayPregunta4", arrayPregunta4);
32
33     return "examen";
34 }
35
36
37 @PostMapping("/enviarprueba")
38 public String guardarNotas(@Validated Notas notas, Errors errores) {
39
40     if (errores.hasErrors()) {
41         // si tenemos errores retornamos a la vista de modificar
42         return "redirect:/examen";
43     }
44
45     notasService.guardar(notas);
46     return "redirect:/notas";
47 }
48
```

Por ultimo tenemos el metodo notas que se encargara de recuperar la información de notas en la base de datos, hacer su respectiva validación con el operador ternario y pasarlo al modelo, también creamos la variable calificación donde le vamos a asignar la nota final al estudiante, cada punto vale 25, para así dar un total de 100.

```
88 @GetMapping("/notas")
89 public String notas(Model model) {
90
91     var estudiante = estudianteService.listarPersonas();
92     String nombreEstudiante = "";
93     for(var n: estudiante){
94         nombreEstudiante = n.getNombre();
95     }
96
97     model.addAttribute("nombreEstudiante", nombreEstudiante);
98
99     var notas = notasService.listarNotas();
100
101     String nota1 = "", nota2="", nota3="", nota4="";
102     for(var p: notas){
103         nota1 = p.getNota1();
104         nota2 = p.getNota2();
105         nota3 = p.getNota3();
106         nota4 = p.getNota4();
107     }
108
109     int puntaje1 = nota1.equals("Java") ? (puntaje1 = 25) : (puntaje1 = 0);
110     int puntaje2 = nota2.equals("James Gosling") ? (puntaje1 = 25) : (puntaje1 = 0);
111     int puntaje3 = nota3.equals("1995") ? (puntaje1 = 25) : (puntaje1 = 0);
112     int puntaje4 = nota4.equals("public class Componente extends Producto") ? (puntaje1 = 25) : (puntaje1 = 0);
113     int calificacion = puntaje1 + puntaje2 + puntaje3 + puntaje4;
114
115     model.addAttribute("puntaje1", puntaje1);
116     model.addAttribute("puntaje2", puntaje2);
117     model.addAttribute("puntaje3", puntaje3);
118     model.addAttribute("puntaje4", puntaje4);
119     model.addAttribute("calificacion", calificacion);
120
121     model.addAttribute("notas", notas);
122
123     //zona ID
124
125     String zona = "";
126     for(var n: estudiante){
127         nombreEstudiante = n.getNombre();
128         zona = n.getZona();
129     }
130
131     ZoneId zonaId = ZoneId.of(zona);
132
133     ComponentEstudiante zonaEstudiante = new ComponentEstudiante();
134     ZonedDateTime zonaDate = zonaEstudiante.obtenerZonaHoraria(zonaId);
135     String zonaparse = zonaDate.toString();
136     model.addAttribute("zonaparse", zonaparse);
137     return "nota";
138 }
```

Paginas



En la página de index lo único destacable es el botón que nos redirige a la pagina de estudiante

```
<!-- Submit button -->
<div class="container-fluid">
  <div class="row">
    <a th:href="@{/agregar}" class="btn btn-primary" style="background-color: #002060; border-color: #002060; margin: 0 auto; max-widt
  </div>
</div>
```

En la pagina de estudiante creamos el formulario y asociamos los campos utilizando thymeleaf al formulario y dando las distintas propiedades necesarias para su correcto funcionamiento

```
<!-- Form del estudiante -->
<form th:action="@{/guardar}" method="post" th:object="${estudiante}">

  <!-- inicio datos de usuario -->

  <div class="row mb-4">
    <div class="col">
      <div class="form-outline">
        <label class="form-label" for="nombre">Nombre</label>
        <input type="text" name="nombre" th:field="**{nombre}"
          class="form-control" placeholder="Ingrese su nombre" required />
      </div>
    </div>
    <div class="col">
      <div class="form-outline">
        <label class="form-label" for="edad">Edad</label>
        <input type="number" name="edad" th:field="**{edad}"
          class="form-control" placeholder="Ingrese su edad" required />
      </div>
    </div>
  </div>

  <div class="row mb-4">
    <div class="col">
      <div class="form-outline">
        <label class="form-label" for="ciudad">Ciudad</label>
        <input type="text" name="ciudad" th:field="**{ciudad}"
          class="form-control" placeholder="Ingrese su ciudad" required />
      </div>
    </div>

    <div class="col">
      <div class="form-outline">
        <label class="form-label">Zona Horaria</label>
        <select class="form-control" id="zona" name="zona">
          <option value="US/Central">(GMT-06:00) Central Time (US - Canada)</option>
          <option value="America/Mexico_City">(GMT-06:00) Guadalajara, Mexico City, Monterrey</option>
          <option value="America/Bogota" selected="">(GMT-05:00) Bogota, Lima, Quito, Rio Branco</option>
          <option value="America/Argentina/Cordoba">(GMT-03:00) America, Argentina, Cordoba</option>
          <option value="Hongkong">(GMT-08:00) Hongkong</option>
        </select>
      </div>
    </div>
  </div>

</div>
```

En la pagina de examen iteramos el arreglo de preguntas y con expresion lenguaje recuperamos los valores

```
<div class="form-check" >
  <div class="pregunta">
    <h5 th:text="{pregunta1}"></h5>
  </div>

  <div th:each="array : ${arrayPregunta1}">
    <input class="form-check-input" type="radio" name="nota1" id="nota1" th:value="{array}" >
    <label class="form-check-label" >
      <span th:text="{array}">opciones1</span>
    </label>
  </div>
</div>

<div class="form-check">
  <div class="pregunta">
    <h5 th:text="{pregunta2}"></h5>
  </div>
  <div th:each="array : ${arrayPregunta2}">
    <input class="form-check-input" type="radio" name="nota2" id="nota2" th:value="{array}" >
    <label class="form-check-label" >
      <span th:text="{array}">opciones2</span>
    </label>
  </div>
</div>

<div class="form-check">
  <div class="pregunta">
    <h5 th:text="{pregunta3}"></h5>
  </div>
  <div th:each="array : ${arrayPregunta3}">
    <input class="form-check-input" type="radio" name="nota3" id="nota3" th:value="{array}" >
    <label class="form-check-label" >
      <span th:text="{array}">opciones3</span>
    </label>
  </div>
</div>

<div class="form-check">
  <div class="pregunta">
    <h5 th:text="{pregunta4}"></h5>
  </div>
  <div th:each="array : ${arrayPregunta4}">
    <input class="form-check-input" type="radio" name="nota4" id="nota4" th:value="{array}" >
    <label class="form-check-label" >
      <span th:text="{array}">opciones4</span>
    </label>
  </div>
</div>
```


Y al final en el html notas recuperamos los valores y los mostramos en un listado

```
<div class="container">
  <!--CABECERO-->
  <div class="row bienvenida">
    <div class="titulo">
      Examen
    </div>
  </div>

  <div>
    <table class="table table-striped">
      <thead class="thead-dark">
        <tr>
          <th>Nombre</th>
          <th>Nota1</th>
          <th>Nota2</th>
          <th>Nota3</th>
          <th>Nota4</th>
          <th>Calificacion</th>
          <th>Zona Horaria</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td th:text="{nombreEstudiante}"></td>
          <td th:text="{puntaje1}">Mostrar nota</td>
          <td>[[{puntaje2}]]</td>
          <td>[[{puntaje3}]]</td>
          <td>[[{puntaje4}]]</td>
          <td>[[{calificacion}]]</td>
          <td>[[{zonaparse}]]</td>
        </tr>
      </tbody>
    </table>
  </div>

  <!-- Submit button -->
  <!-- Submit button -->
  <div class="container-fluid">
    <div class="row">
      <a th:href="@{...}" class="btn btn-primary" style="background-color: #002060; border-color: #002060; margin: 0 auto; max-width: 100px;">
    </div>
  </div>
</div>
```

Archivo properties

El jpa y la configuración a la base de datos esta configurada de la siguiente manera:

Para este caso la base de datos no tiene contraseña, por lo tanto no fue necesario especificarlo

```
1
2 spring.main.banner-mode=off
3 spring.thymeleaf.cache=false
4 #Mysql conexion
5 spring.datasource.url=jdbc:mysql://localhost/fonyoubd?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true
6 spring.datasource.username=root
7 #spring.datasource.password=
8 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
9 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
10 #Mostrar sql
11 spring.jpa.properties.hibernate.format_sql=true
12 #logging.level.org.hibernate.SQL=DEBUG
13 #logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
14 spring.jpa.show-sql=true
```

Esto es todo lo que realice para el ejercicio, espero sea de su agrado y les haya parecido bastante agradable la implementación, muchas gracias :D