Evidencia de conocimiento

- Fecha: 01-03-2024
- Hora: 13:00 - 15:00
- Entrega Moodle en el tiempo establecido.

Backend

Descripción - crear poryecto con la siguiente estructura:

- Nombre: PrimerNombrePrimerApellido
- COM: sena
- Dependencias: Spring Boot DevTools, Spring Web, Spring Data JPA, MySQL Driver
- Java: (Sugerida LTS)
- Spring Boot: (Sugerida LTS)

Datos de configuración

- server.port=
- spring.jpa.hibernate.ddl-auto =
- spring.datasource.url = jdbc:mysql:
- spring.datasource.username =
- spring.datasource.password =

- Nota: Usar puerto 909+(último digito de su documento) para el servidor y el 3306 para el motor de base de datos.
- Ejemplo de ruta driver SQL: jdbc:mysql://localhost:3306/nameBD

Paquetes de aplicación con respectivos archivos (PascalCase):

Entities

- Customer
- Product
- CustomerProduct

IRepository

- ICustomerRepository

- IProductRepository
- ICustomerProductRepository

IService

- ICustomerService
- IProductService
- ICustomerProductService

Service

- CustomerService
- ProductService
- CustomerProductService

Controller

- CustomerController
- ProductController
- CustomerProductController

Base de datos

Datos de la base de datos

- User: u342060465_ev_api
- Base de Datos: u342060465_ev_api
- Password = 4=cvqys$vW;R
- IP = 149.100.155.52

Necesidadad

Entity Customer with the following attributes "customer_document_student":

- id: BIGINT (primary key, auto-incremental)
- name: VARCHAR(50)
- email: VARCHAR(50)
- phone: VARCHAR(20)
- address: VARCHAR(100)

Entity Product with the following attributes "product_document_student":

- id: BIGINT (primary key, auto-incremental)
- name: VARCHAR(50)
- description: TEXT

Entity CustomerProduct representing products associated with a customer:

- id: BIGINT (primary key, auto-incremental)
- customer_id: BIGINT (foreign key)
- product_id: BIGINT (foreign key, unique constraint)
- balance: DECIMAL(10,2)

Audit fields (all entities)

- state
- createdAt
- updatedAt
- deletedAt

API

Datos de la API

- api/customer
- api/product
- api/customer-product
- Body: raw, JSON
- Métodos:
    - GET: FindAll, FindById
    - POST: Save
    - PUT: Update
    - PUT: DeleteLogical
    - DELETE: DeleteById

Documentación de la API

Colección: customer

- Carpeta: customer
    - GET: FindAll
    - GET: FindById
    - POST: Save
    - PUT: Update
    - PUT: DeleteLogical
    - DELETE: DeleteById

Colección: product

- Carpeta: product
    - GET: FindAll
    - GET: FindById
    - POST: Save
    - PUT: Update
    - PUT: DeleteLogical
    - DELETE: DeleteById

Colección: customer-product

- Carpeta: customer-product
    - GET: FindAll
    - GET: FindById
    - POST: Save
    - PUT: Update
    - PUT: DeleteLogical
    - DELETE: DeleteById

Ejemplo de body

- Si los atributos de "Departamento" son code y name, de la api api/estate debe ser:
- ```
  {
      "code": "12",
      "name": "Huila"
  }
  ```
- Si los atrubitos de "City" son code, name y estateId, de la api api/city, el body debe ser:
- ```
  {
      "code": "121",
      "name": "Neiva",
      "estateId": {
          "id": 1
      }
  }
  ```

Entregar

Backend: Código fuente

Base de datos: Script de creación de la base de datos

Documentación: Documentación de la API con postman (archivo json)

Ejemplo de código

Entity

```
@Entity
@Table(name="materia_estudiante")
public class MateriaEstudiante {

	@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;

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

	@ManyToOne(fetch = FetchType.EAGER, optional = false)
    @JoinColumn(name = "estudiante_id", nullable = false, unique =
true)
    public Estudiante estudianteId;

	// Encapsulamiento

}
```

IRepository

```
    @Repository
```

```
    public interface IMateriaEstudianteRepository extends
JpaRepository<MateriaEstudiante, Long>{

    }
```

IService

```
    public interface IMateriaEstudianteService {
        List<MateriaEstudiante> all();
        public Optional<MateriaEstudiante> findById(Long id);
        public MateriaEstudiante save(MateriaEstudiante
materiaEstudiante);
        public void update(MateriaEstudiante materiaEstudiante,
Long id);
        public void delete(Long id);
    }
```

Servic

```
    @Service
    public class MateriaEstudianteService implements
IMateriaEstudianteService {

        @Autowired
        private IMateriaEstudianteRepository
iMateriaEstudianteRepository;

        @Override
        public List<MateriaEstudiante> all() {
            return iMateriaEstudianteRepository.findAll();
        }

        @Override
        public Optional<MateriaEstudiante> findById(Long id) {
            return iMateriaEstudianteRepository.findById(id);
        }

        @Override
        public MateriaEstudiante save(MateriaEstudiante
materiaEstudiante) {
            return
iMateriaEstudianteRepository.save(materiaEstudiante);
```

```
        }

        @Override
        public void update(MateriaEstudiante materiaEstudiante,
Long id) {
            //validar si existe.
            Optional<MateriaEstudiante> op =
iMateriaEstudianteRepository.findById(id);

            if(op.isEmpty()){
                System.out.println("Dato no encontrado");
            }else{
                //Crear nuevo objeto que va a contener los datos
que se van actualizar
                MateriaEstudiante materiaEstudianteUpdate =
op.get();

materiaEstudianteUpdate.setMateria(materiaEstudiante.getMateria());

materiaEstudianteUpdate.setEstudianteId(materiaEstudiante.getEstudi
anteId());

                //Actualizar el objeto

iMateriaEstudianteRepository.save(materiaEstudianteUpdate);
            }
        }

        @Override
        public void delete(Long id) {
            iMateriaEstudianteRepository.deleteById(id);
        }
    }
```

Controller

```
    @CrossOrigin(origins = "*")
    @RestController
    @RequestMapping("v1/api/MateriaEstudiante")
    public class MateriaEstudianteController {

        @Autowired
```

```
        private IMateriaEstudianteService service;

        @GetMapping()
        public List<MateriaEstudiante> all() {
            return service.all();
        }

        @GetMapping("{id}")
        public Optional<MateriaEstudiante> show(@PathVariable Long
id) {
            return service.findById(id);
        }

        @PostMapping
        @ResponseStatus(code = HttpStatus.CREATED)
        public MateriaEstudiante save(@RequestBody
MateriaEstudiante materiaEstudiante) {
            return service.save(materiaEstudiante);
        }

        @PutMapping("{id}")
        @ResponseStatus(code = HttpStatus.NO_CONTENT)
        public void update(@RequestBody MateriaEstudiante
materiaEstudiante, @PathVariable Long id) {
            service.update(materiaEstudiante, id);
        }

        @DeleteMapping("{id}")
        @ResponseStatus(code = HttpStatus.NO_CONTENT)
        public void delete(@PathVariable Long id) {
            service.delete(id);
        }

    }
```

Muchos exitos en el desarrollo de la prueba.