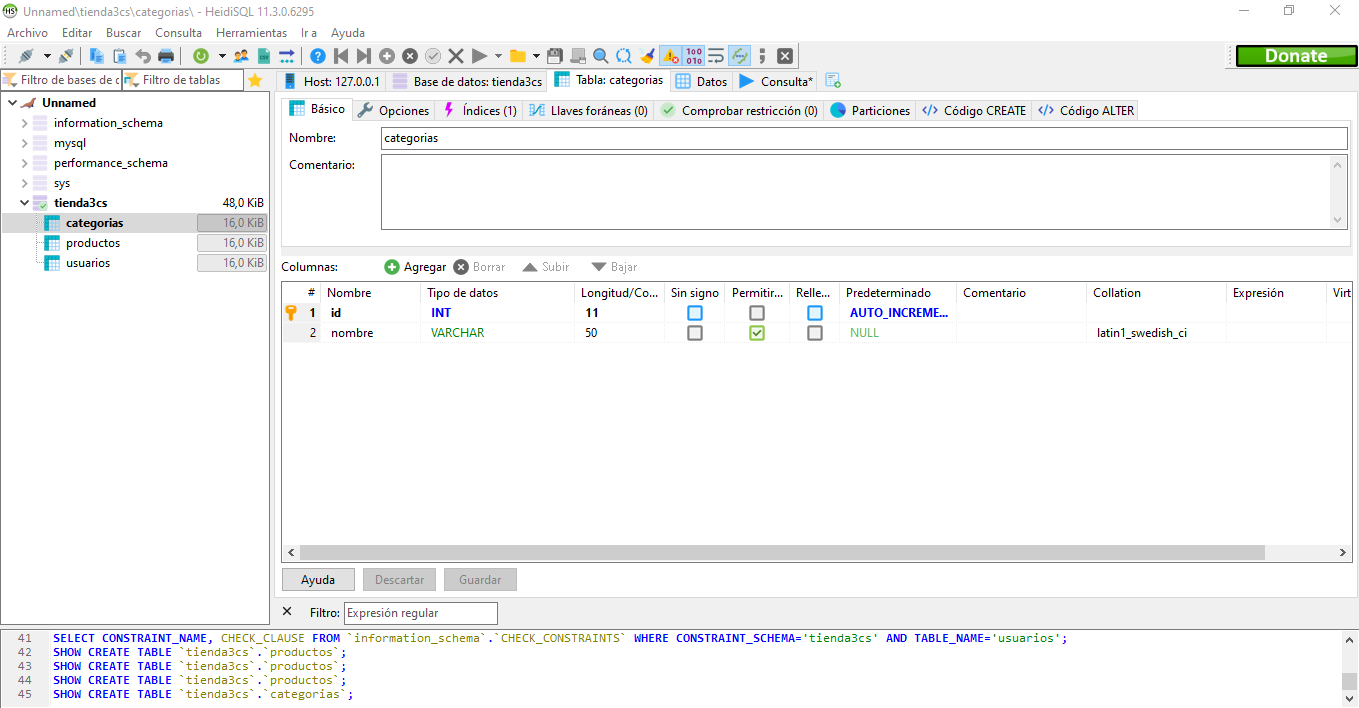
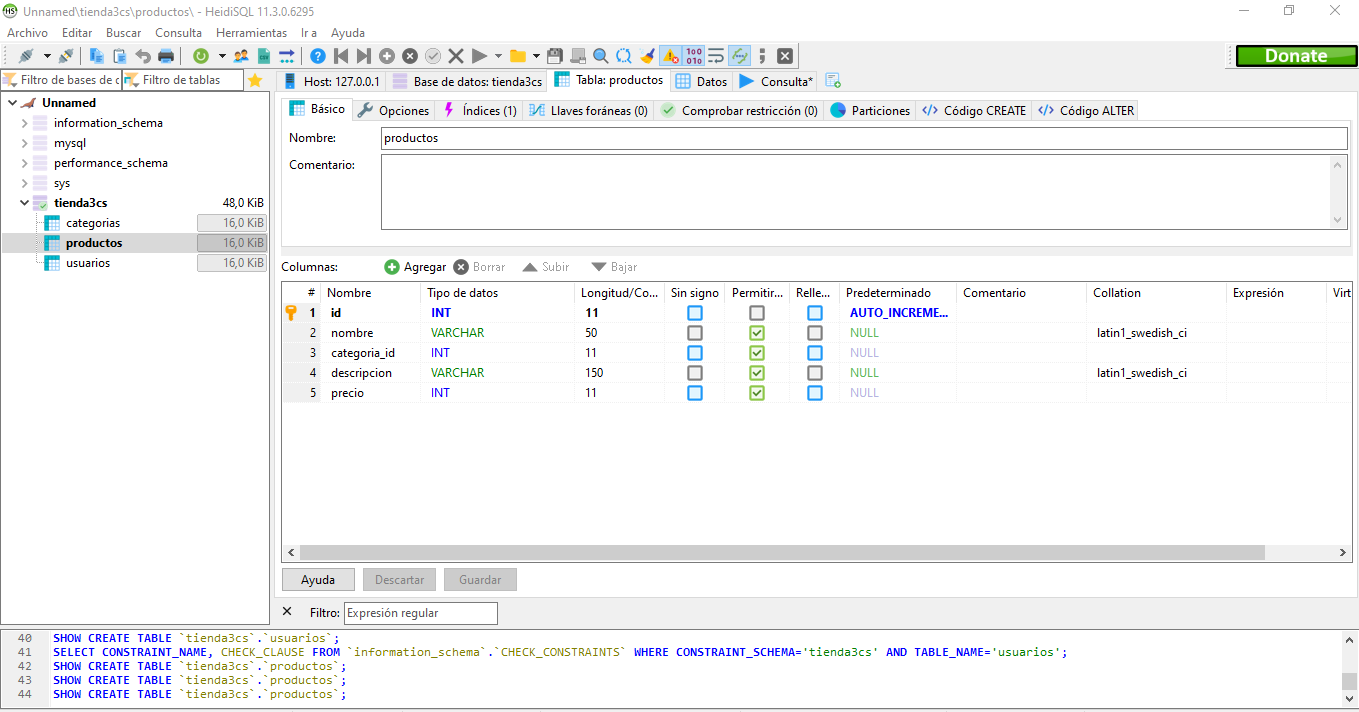
|  |
| --- |
| **SPRINT 2: Repositorio de Código - Diseño Base de Datos** |

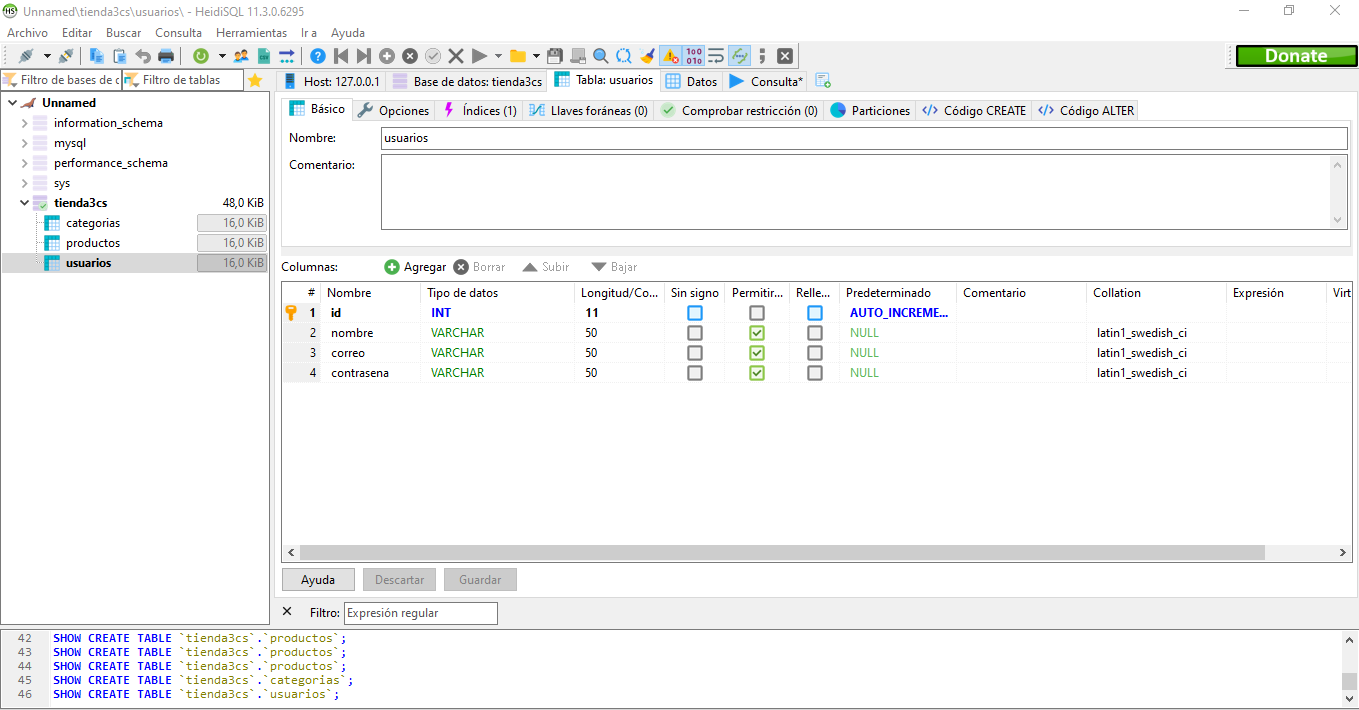
|  |  |
| --- | --- |
| **Identificación Proyecto** | |
| Nombre Proyecto: | Tienda Camisetas G2 |
| Número Equipo: |  |
| **Integrantes del equipo** | |
| Rol  (Líder-Desarrollador – Cliente) | Nombre |
| **Scrum Master** | Oscar Ramírez |
| **Product Owner** | Erick Marcial Machacado Rizo |
| **Development Team** | Jenny Moreno |

|  |
| --- |
| **Diseño de la Base de Datos (Proceso de normalización)** |

Como evidencia del Diseño de la Base de Datos.







|  |
| --- |
| **Esquema de la Base de Datos (Código SQL)** |

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

DROP DATABASE IF EXISTS `tienda3cs`;

CREATE DATABASE IF NOT EXISTS `tienda3cs` /\*!40100 DEFAULT CHARACTER SET latin1 \*/;

USE `tienda3cs`;

DROP TABLE IF EXISTS `categorias`;

CREATE TABLE IF NOT EXISTS `categorias` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`nombre` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `productos`;

CREATE TABLE IF NOT EXISTS `productos` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`nombre` varchar(50) DEFAULT NULL,

`categoria\_id` int(11) DEFAULT NULL,

`descripcion` varchar(150) DEFAULT NULL,

`precio` int(11) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `usuarios`;

CREATE TABLE IF NOT EXISTS `usuarios` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`nombre` varchar(50) DEFAULT NULL,

`correo` varchar(50) DEFAULT NULL,

`contrasena` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=latin1;

/\*!40101 SET SQL\_MODE=IFNULL(@OLD\_SQL\_MODE, '') \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=IFNULL(@OLD\_FOREIGN\_KEY\_CHECKS, 1) \*/;

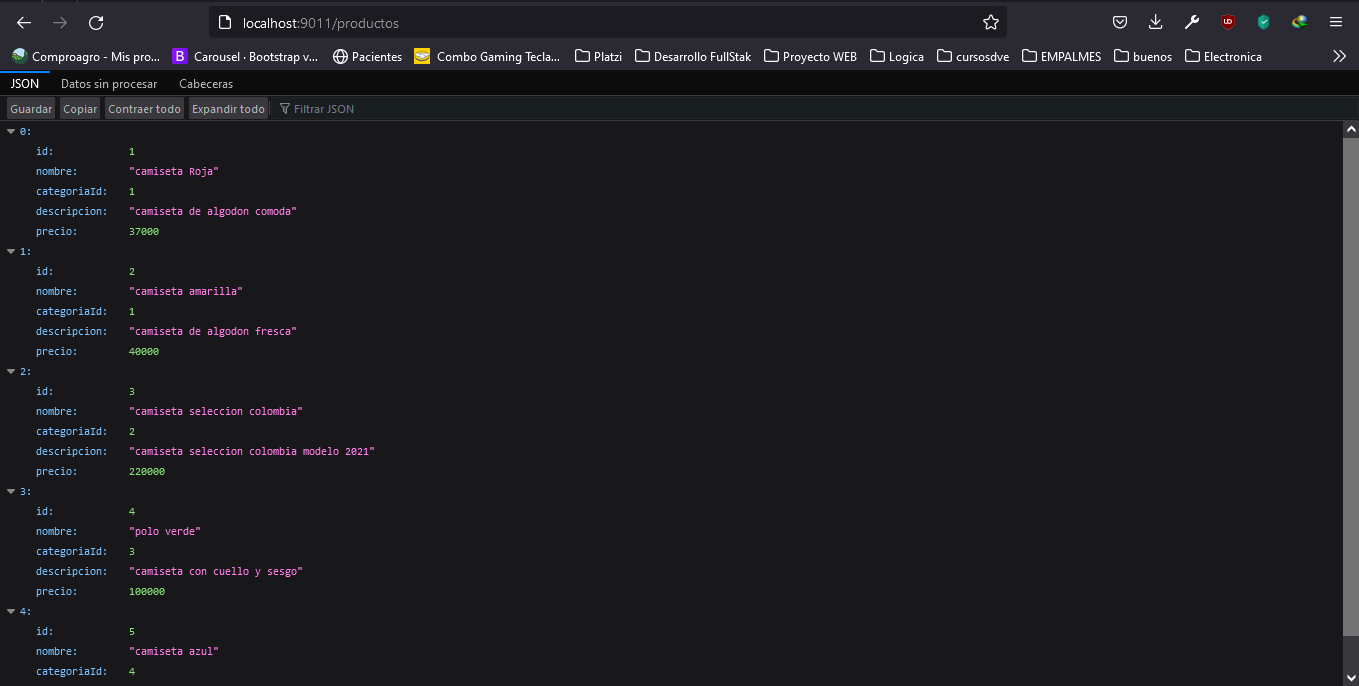
/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40111 SET SQL\_NOTES=IFNULL(@OLD\_SQL\_NOTES, 1) \*/;

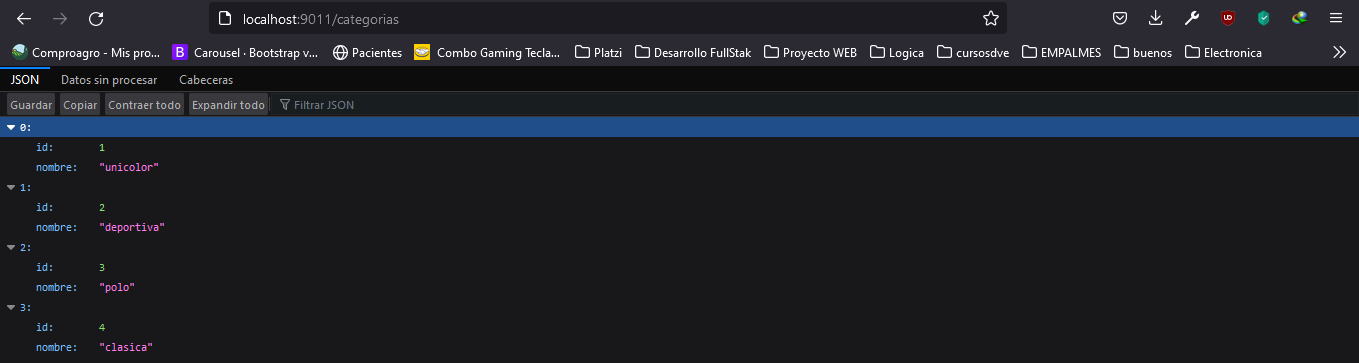
|  |
| --- |
| **Evidencia de la API** |

Como evidencia del listado de productos, categorías y usuarios, Usando el verbo GET

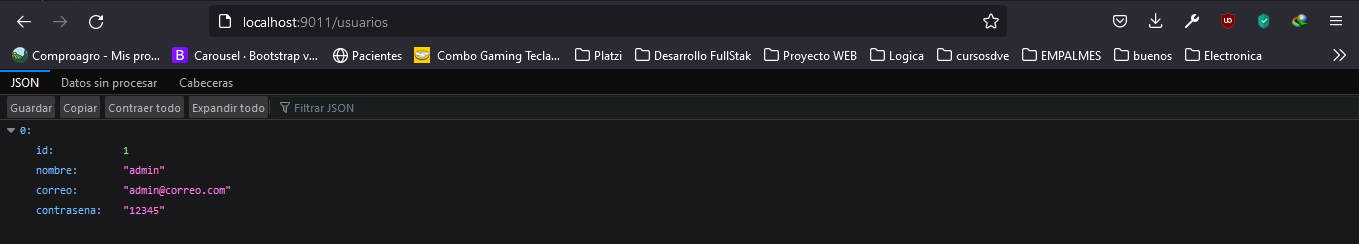
Listado de Productos



Listado de Categorías



Listado de Usuarios



|  |
| --- |
| **Esquema del APIs (Código Java)** |

Como evidencia del listado de productos, categorías y usuarios, en un navegador.

**Listado de Productos**

*/\**

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license*

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template*

*\*/*

*package com.example.demo.Controllers;*

*import com.example.demo.Controllers.exceptions.NonexistentEntityException;*

*import com.example.demo.Models.Productos;*

*import java.io.Serializable;*

*import java.util.List;*

*import javax.persistence.EntityManager;*

*import javax.persistence.EntityManagerFactory;*

*import javax.persistence.Query;*

*import javax.persistence.EntityNotFoundException;*

*import javax.persistence.criteria.CriteriaQuery;*

*import javax.persistence.criteria.Root;*

*import org.springframework.web.bind.annotation.GetMapping;*

*import org.springframework.web.bind.annotation.RequestMapping;*

*import org.springframework.web.bind.annotation.RestController;*

*@RestController*

*@RequestMapping("/productos")*

*public class ProductosJpaController implements Serializable {*

*public ProductosJpaController(EntityManagerFactory emf) {*

*this.emf = emf;*

*}*

*private EntityManagerFactory emf = null;*

*public EntityManager getEntityManager() {*

*return emf.createEntityManager();*

*}*

*public void create(Productos productos) {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*em.persist(productos);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void edit(Productos productos) throws NonexistentEntityException, Exception {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*productos = em.merge(productos);*

*em.getTransaction().commit();*

*} catch (Exception ex) {*

*String msg = ex.getLocalizedMessage();*

*if (msg == null || msg.length() == 0) {*

*Integer id = productos.getId();*

*if (findProductos(id) == null) {*

*throw new NonexistentEntityException("The productos with id " + id + " no longer exists.");*

*}*

*}*

*throw ex;*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void destroy(Integer id) throws NonexistentEntityException {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*Productos productos;*

*try {*

*productos = em.getReference(Productos.class, id);*

*productos.getId();*

*} catch (EntityNotFoundException enfe) {*

*throw new NonexistentEntityException("The productos with id " + id + " no longer exists.", enfe);*

*}*

*em.remove(productos);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*@GetMapping()*

*public List<Productos> findProductosEntities() {*

*return findProductosEntities(true, -1, -1);*

*}*

*public List<Productos> findProductosEntities(int maxResults, int firstResult) {*

*return findProductosEntities(false, maxResults, firstResult);*

*}*

*private List<Productos> findProductosEntities(boolean all, int maxResults, int firstResult) {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*cq.select(cq.from(Productos.class));*

*Query q = em.createQuery(cq);*

*if (!all) {*

*q.setMaxResults(maxResults);*

*q.setFirstResult(firstResult);*

*}*

*return q.getResultList();*

*} finally {*

*em.close();*

*}*

*}*

*public Productos findProductos(Integer id) {*

*EntityManager em = getEntityManager();*

*try {*

*return em.find(Productos.class, id);*

*} finally {*

*em.close();*

*}*

*}*

*public int getProductosCount() {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*Root<Productos> rt = cq.from(Productos.class);*

*cq.select(em.getCriteriaBuilder().count(rt));*

*Query q = em.createQuery(cq);*

*return ((Long) q.getSingleResult()).intValue();*

*} finally {*

*em.close();*

*}*

*}*

*}*

**Listado de Categorías**

*/\**

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license*

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template*

*\*/*

*package com.example.demo.Controllers;*

*import com.example.demo.Controllers.exceptions.NonexistentEntityException;*

*import com.example.demo.Models.Categorias;*

*import java.io.Serializable;*

*import java.util.List;*

*import javax.persistence.EntityManager;*

*import javax.persistence.EntityManagerFactory;*

*import javax.persistence.Query;*

*import javax.persistence.EntityNotFoundException;*

*import javax.persistence.criteria.CriteriaQuery;*

*import javax.persistence.criteria.Root;*

*import org.springframework.web.bind.annotation.GetMapping;*

*import org.springframework.web.bind.annotation.RequestMapping;*

*import org.springframework.web.bind.annotation.RestController;*

*@RestController*

*@RequestMapping("/categorias")*

*public class CategoriasJpaController implements Serializable {*

*public CategoriasJpaController(EntityManagerFactory emf) {*

*this.emf = emf;*

*}*

*private EntityManagerFactory emf = null;*

*public EntityManager getEntityManager() {*

*return emf.createEntityManager();*

*}*

*public void create(Categorias categorias) {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*em.persist(categorias);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void edit(Categorias categorias) throws NonexistentEntityException, Exception {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*categorias = em.merge(categorias);*

*em.getTransaction().commit();*

*} catch (Exception ex) {*

*String msg = ex.getLocalizedMessage();*

*if (msg == null || msg.length() == 0) {*

*Integer id = categorias.getId();*

*if (findCategorias(id) == null) {*

*throw new NonexistentEntityException("The categorias with id " + id + " no longer exists.");*

*}*

*}*

*throw ex;*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void destroy(Integer id) throws NonexistentEntityException {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*Categorias categorias;*

*try {*

*categorias = em.getReference(Categorias.class, id);*

*categorias.getId();*

*} catch (EntityNotFoundException enfe) {*

*throw new NonexistentEntityException("The categorias with id " + id + " no longer exists.", enfe);*

*}*

*em.remove(categorias);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*@GetMapping()*

*public List<Categorias> findCategoriasEntities() {*

*return findCategoriasEntities(true, -1, -1);*

*}*

*public List<Categorias> findCategoriasEntities(int maxResults, int firstResult) {*

*return findCategoriasEntities(false, maxResults, firstResult);*

*}*

*private List<Categorias> findCategoriasEntities(boolean all, int maxResults, int firstResult) {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*cq.select(cq.from(Categorias.class));*

*Query q = em.createQuery(cq);*

*if (!all) {*

*q.setMaxResults(maxResults);*

*q.setFirstResult(firstResult);*

*}*

*return q.getResultList();*

*} finally {*

*em.close();*

*}*

*}*

*public Categorias findCategorias(Integer id) {*

*EntityManager em = getEntityManager();*

*try {*

*return em.find(Categorias.class, id);*

*} finally {*

*em.close();*

*}*

*}*

*public int getCategoriasCount() {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*Root<Categorias> rt = cq.from(Categorias.class);*

*cq.select(em.getCriteriaBuilder().count(rt));*

*Query q = em.createQuery(cq);*

*return ((Long) q.getSingleResult()).intValue();*

*} finally {*

*em.close();*

*}*

*}*

*}*

**Listado de Usuarios**

*/\**

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license*

*\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template*

*\*/*

*package com.example.demo.Controllers;*

*import com.example.demo.Controllers.exceptions.NonexistentEntityException;*

*import com.example.demo.Models.Usuarios;*

*import java.io.Serializable;*

*import java.util.List;*

*import javax.persistence.EntityManager;*

*import javax.persistence.EntityManagerFactory;*

*import javax.persistence.Query;*

*import javax.persistence.EntityNotFoundException;*

*import javax.persistence.criteria.CriteriaQuery;*

*import javax.persistence.criteria.Root;*

*import org.springframework.web.bind.annotation.GetMapping;*

*import org.springframework.web.bind.annotation.RequestMapping;*

*import org.springframework.web.bind.annotation.RestController;*

*@RestController*

*@RequestMapping("/usuarios")*

*public class UsuariosJpaController implements Serializable {*

*public UsuariosJpaController(EntityManagerFactory emf) {*

*this.emf = emf;*

*}*

*private EntityManagerFactory emf = null;*

*public EntityManager getEntityManager() {*

*return emf.createEntityManager();*

*}*

*public void create(Usuarios usuarios) {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*em.persist(usuarios);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void edit(Usuarios usuarios) throws NonexistentEntityException, Exception {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*usuarios = em.merge(usuarios);*

*em.getTransaction().commit();*

*} catch (Exception ex) {*

*String msg = ex.getLocalizedMessage();*

*if (msg == null || msg.length() == 0) {*

*Integer id = usuarios.getId();*

*if (findUsuarios(id) == null) {*

*throw new NonexistentEntityException("The usuarios with id " + id + " no longer exists.");*

*}*

*}*

*throw ex;*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*public void destroy(Integer id) throws NonexistentEntityException {*

*EntityManager em = null;*

*try {*

*em = getEntityManager();*

*em.getTransaction().begin();*

*Usuarios usuarios;*

*try {*

*usuarios = em.getReference(Usuarios.class, id);*

*usuarios.getId();*

*} catch (EntityNotFoundException enfe) {*

*throw new NonexistentEntityException("The usuarios with id " + id + " no longer exists.", enfe);*

*}*

*em.remove(usuarios);*

*em.getTransaction().commit();*

*} finally {*

*if (em != null) {*

*em.close();*

*}*

*}*

*}*

*@GetMapping()*

*public List<Usuarios> findUsuariosEntities() {*

*return findUsuariosEntities(true, -1, -1);*

*}*

*public List<Usuarios> findUsuariosEntities(int maxResults, int firstResult) {*

*return findUsuariosEntities(false, maxResults, firstResult);*

*}*

*private List<Usuarios> findUsuariosEntities(boolean all, int maxResults, int firstResult) {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*cq.select(cq.from(Usuarios.class));*

*Query q = em.createQuery(cq);*

*if (!all) {*

*q.setMaxResults(maxResults);*

*q.setFirstResult(firstResult);*

*}*

*return q.getResultList();*

*} finally {*

*em.close();*

*}*

*}*

*public Usuarios findUsuarios(Integer id) {*

*EntityManager em = getEntityManager();*

*try {*

*return em.find(Usuarios.class, id);*

*} finally {*

*em.close();*

*}*

*}*

*public int getUsuariosCount() {*

*EntityManager em = getEntityManager();*

*try {*

*CriteriaQuery cq = em.getCriteriaBuilder().createQuery();*

*Root<Usuarios> rt = cq.from(Usuarios.class);*

*cq.select(em.getCriteriaBuilder().count(rt));*

*Query q = em.createQuery(cq);*

*return ((Long) q.getSingleResult()).intValue();*

*} finally {*

*em.close();*

*}*

*}*

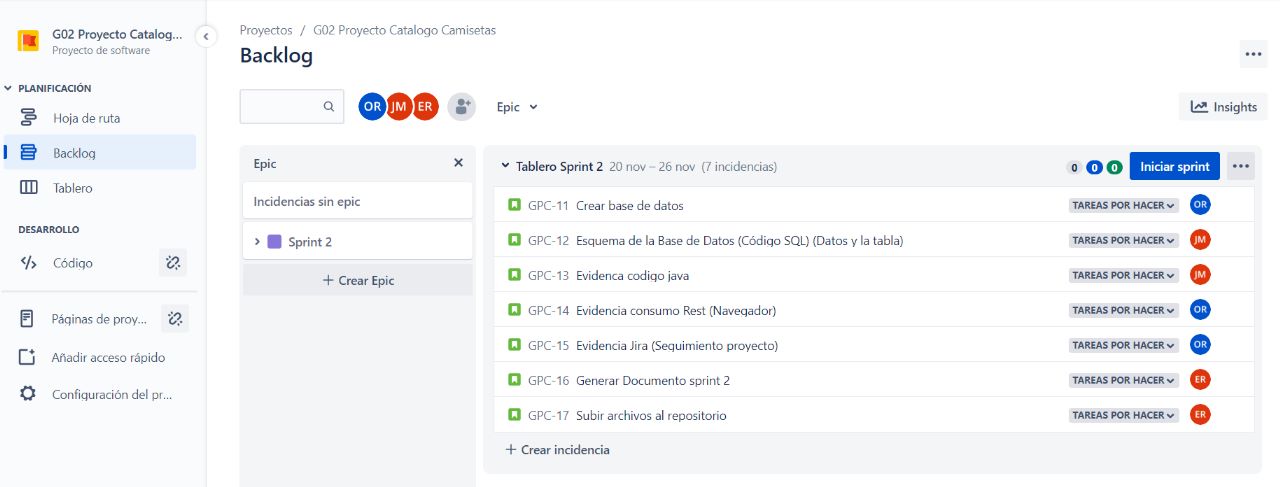
*}*

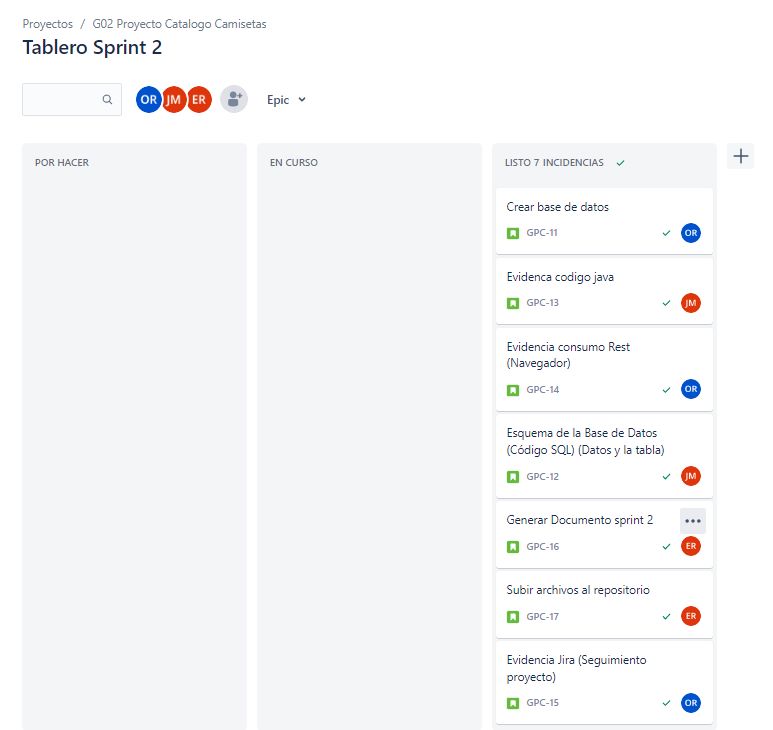
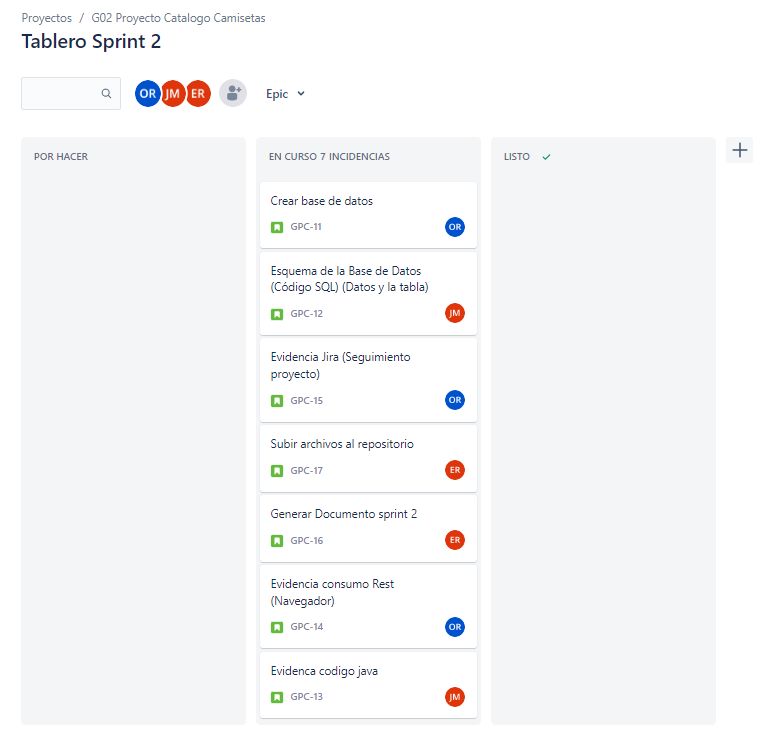
|  |
| --- |
| **Evidencia JIRA (Seguimiento del proyecto)** |

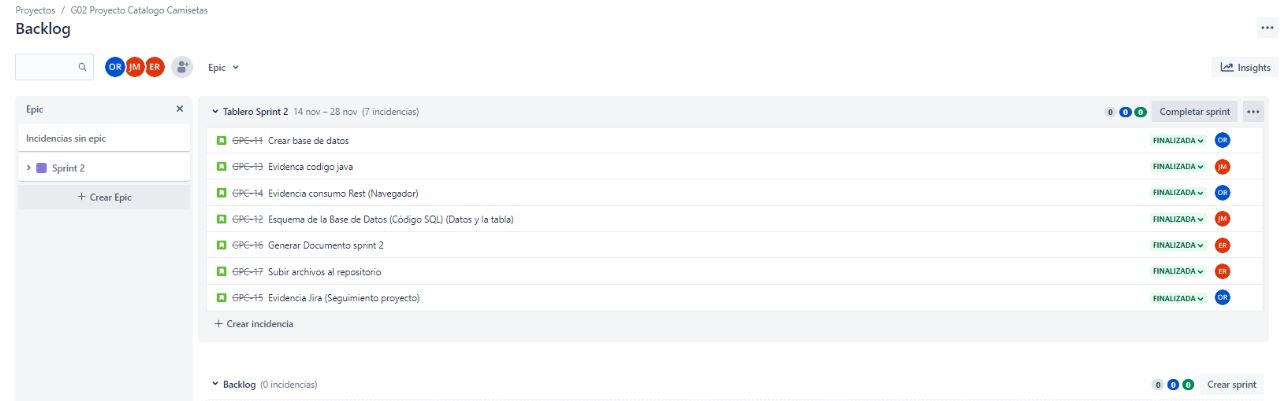
Como evidencia del seguimiento del proyecto con la metodología ágil SCRUM, utilizando el software JIRA, se debe presentar capturas de pantalla donde se visualice la ejecución de los Sprint con las historias de usuario relacionadas con el repositorio de código y el diseño de la base de datos.

Crear segundo sprint 2, con mínimo 6 tareas.

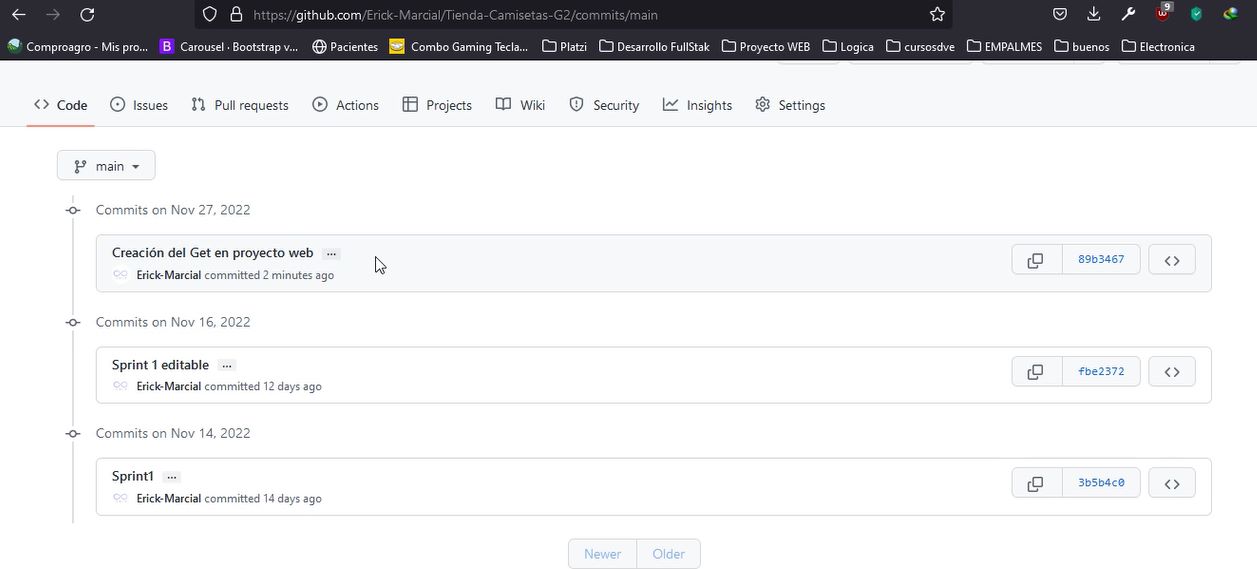
<https://github.com/Erick-Marcial/Tienda-Camisetas-G2>

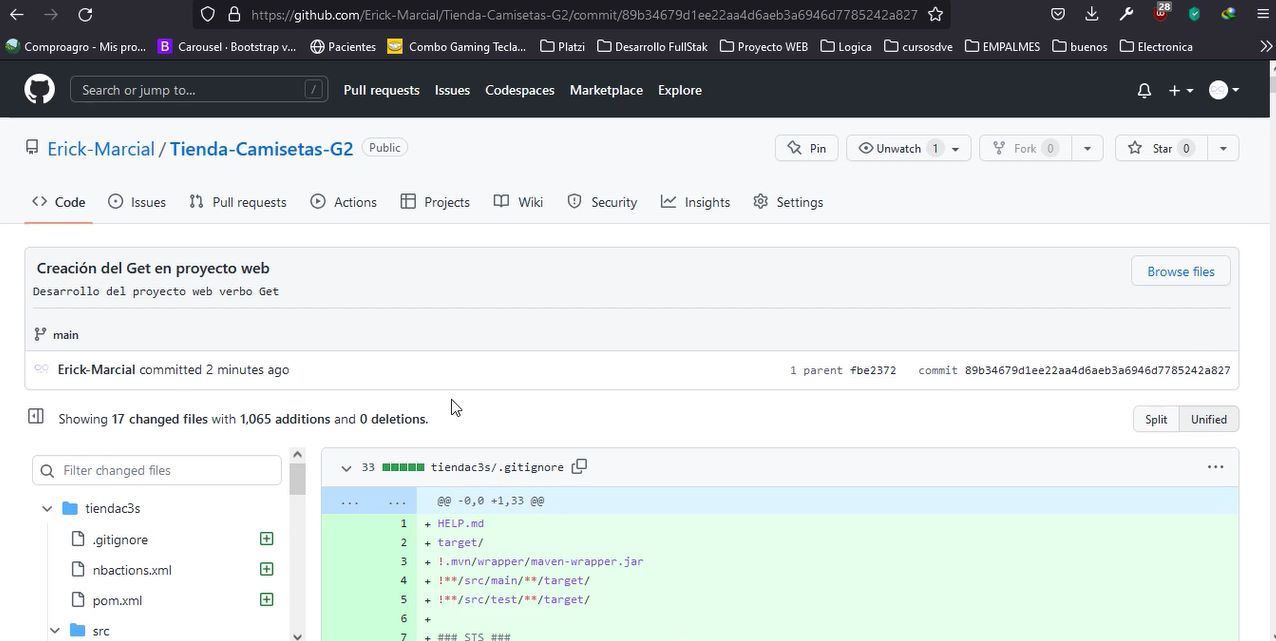












|  |
| --- |
| **Evidencias de las Reuniones de Equipo** |

Como evidencia de las reuniones que efectúa el equipo del proyecto, presentar capturas de pantalla de las reuniones efectuadas y si lo consideran pertinente algunas actas de las reuniones.

