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

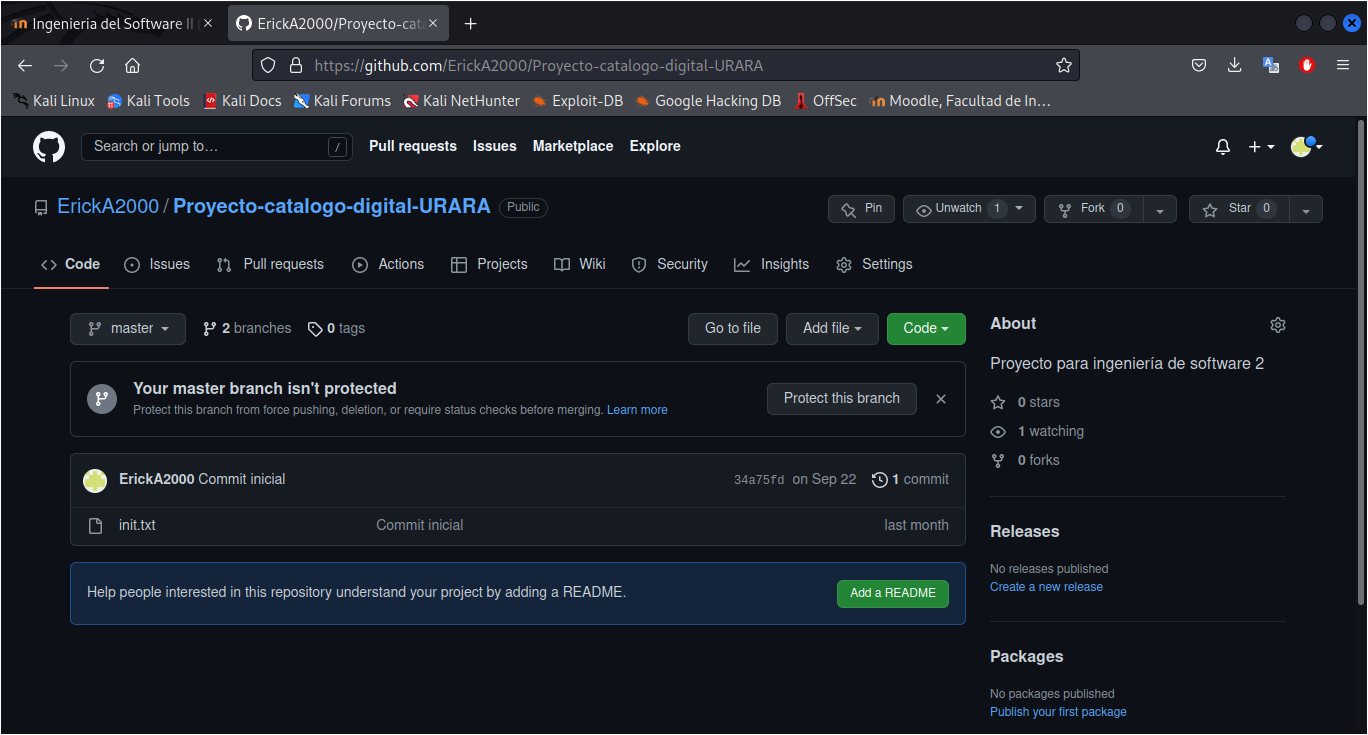
|  |  |
| --- | --- |
| **Identificación Proyecto** | |
| Nombre Proyecto: | Desarrollo de un catálogo digital para ropa de niña urara |
| Número Equipo: | 4 |
| **Integrantes del equipo** | |
| Rol  (Líder-Desarrollador – Cliente) | Nombre |
| Product Owner | Monica Lopez |
| Scrum Master | Erick Arévalo |
| Scrum Team | Jhon Carrascal |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| **Repositorio de Código GitLab o GitHub** |

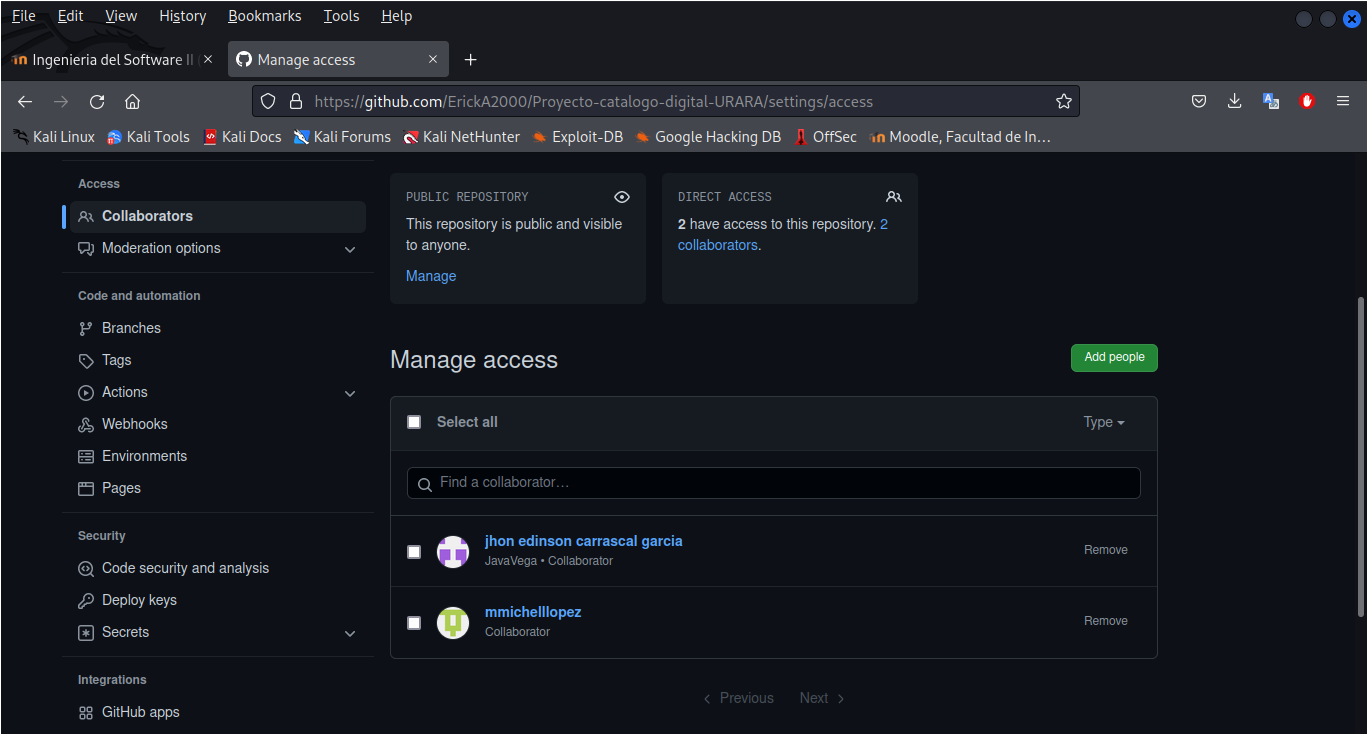
Como evidencia del repositorio de código, creado con GitLab o GitHub, además de la URL del repositorio, se debe presentar capturas de pantalla donde se visualicen aspectos:

Enlace repositorio: https://github.com/ErickA2000/Proyecto-catalogo-digital-URARA.git

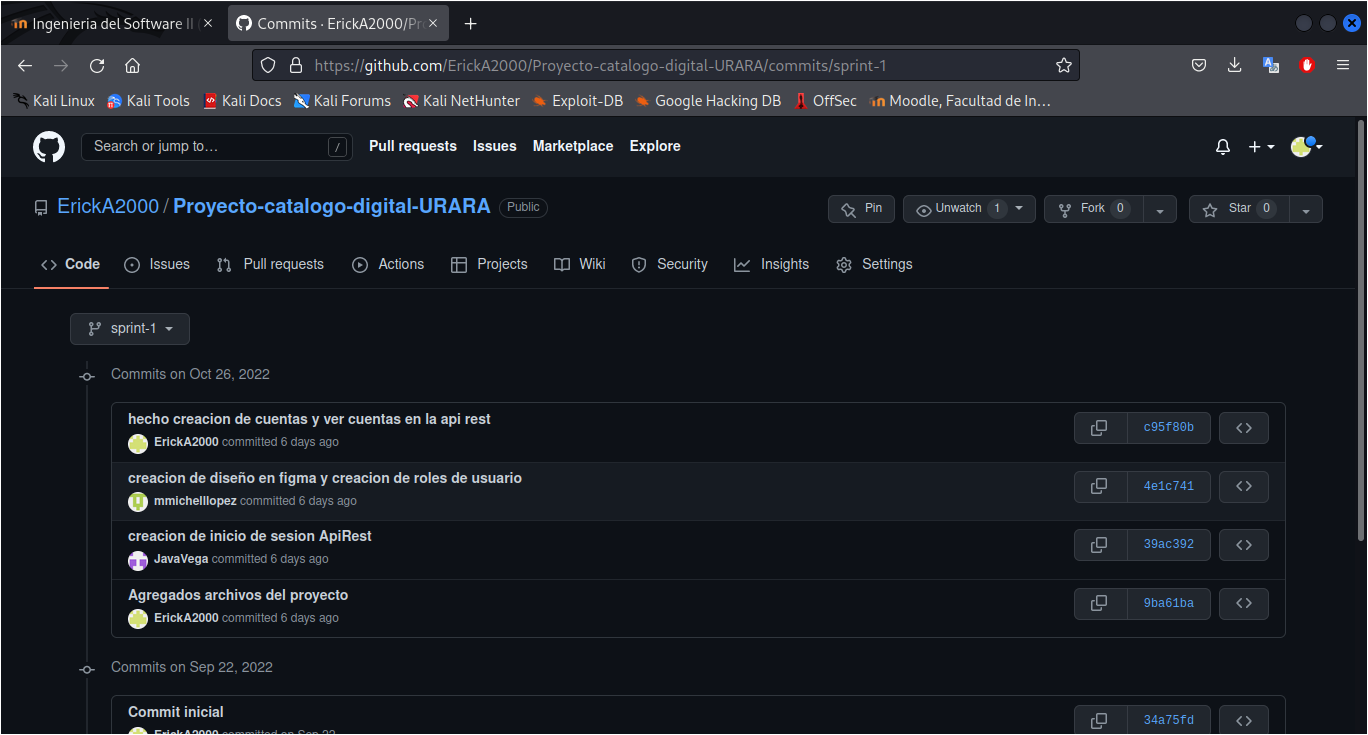
* Creación del proyecto del repositorio.



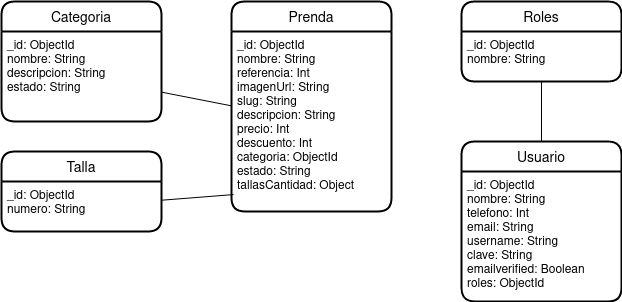
* Integrantes del equipo invitados.



* Evidencia de la realización de alguna actualización (commit), donde se visualice la actualización y el historial de actualizaciones (Versiones)



|  |
| --- |
| **Diseño de la Base de Datos no SQL** |



|  |
| --- |
| **Esquema de la Base de Datos no SQL** |

* **Categoria**

const categoriaSchema = new Schema({

nombre: String,

descripcion: String,

estado: String

},{

timestamps: true,

versionKey: false

});

* **Prenda**

const tallaCantidadSchema = new Schema({

talla: {

ref: "Talla",

type: String

},

cantidad: Number

},{

\_id: false,

versionKey: false

})

const prendaSchema = new Schema({

nombre: String,

referencia: {

type: Number,

unique: true

},

slug: String,

imagenUrl: {

type: Schema.Types.Array

},

descripcion: String,

tallasCantidad: [tallaCantidadSchema],

precio: Number,

descuento: Number,

estado: String,

categoria: [{

ref: "Categorias",

type: Schema.Types.ObjectId

}]

},{

timestamps: true,

versionKey: false

})

* **Roles**

const rolesSchema = new Schema({

nombre: String

},{

versionKey: false

})

* **Talla**

const tallaSchema = new Schema({

numero: String

},{

versionKey: false

})

* **Usuario**

const usuarioSchema = new Schema<IUser, IUserModel>({

nombre: {

type: String

},

telefono: {

type: Number

},

email: {

type: String,

unique: true,

required: true

},

username: {

type: String,

unique: true

},

clave: {

type: String,

required: true

},

emailverified: {

type: Boolean,

default: false

},

roles: [{

ref: "Role",

type: Schema.Types.ObjectId

}]

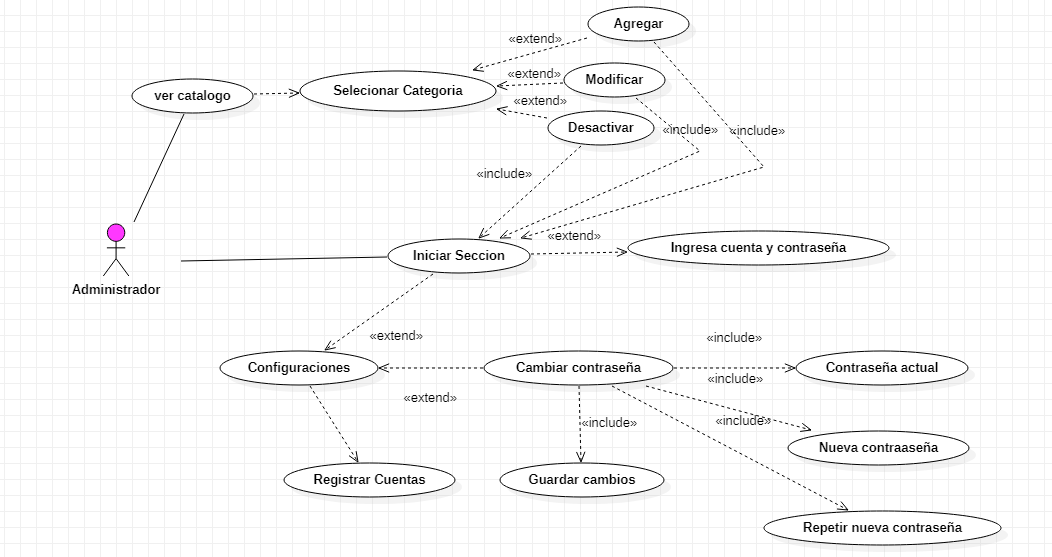
},{

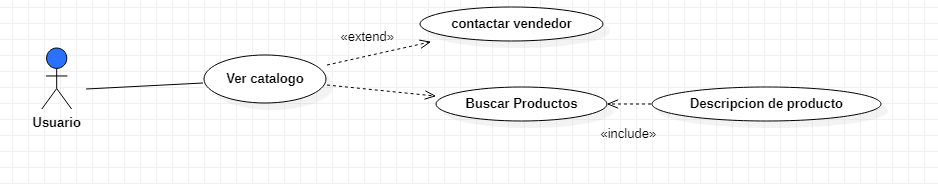
timestamps: true,

versionKey: false

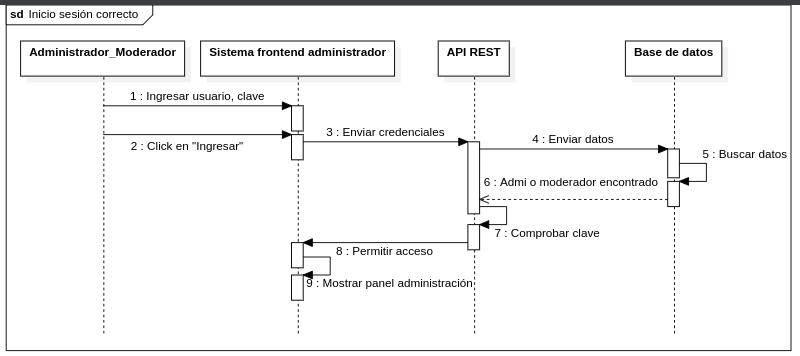
})

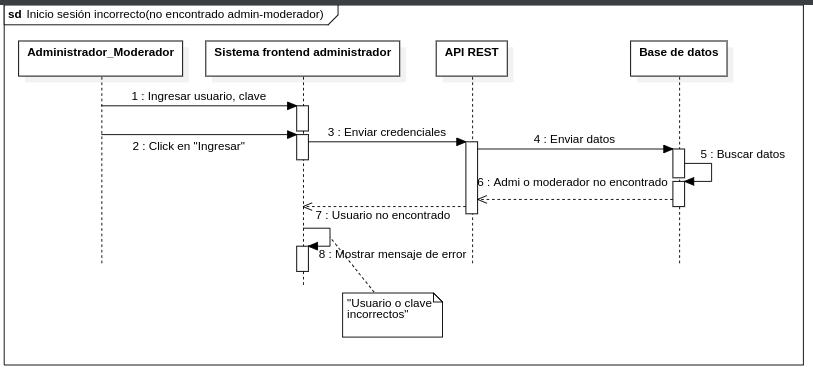
|  |
| --- |
| **Diagrama casos de uso** |

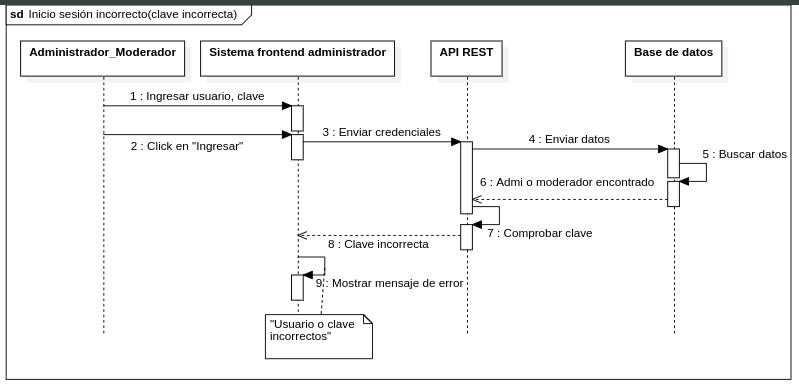
Figure 1: Diagrama casos de uso administrador

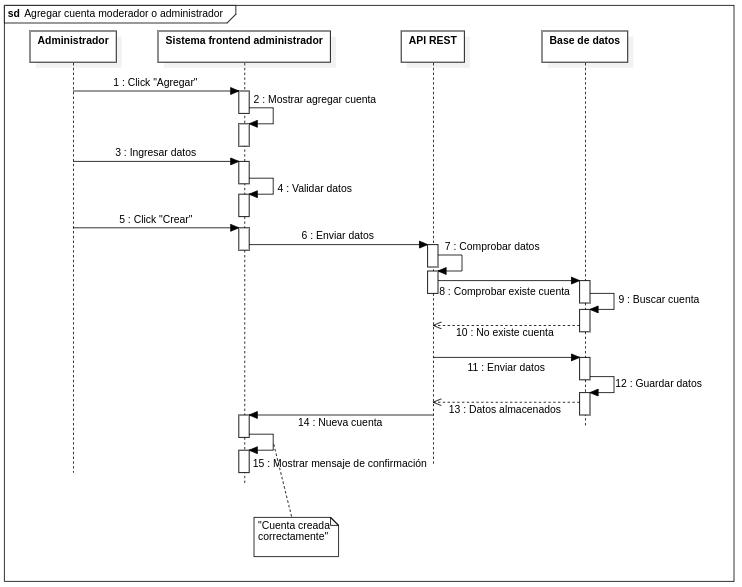
Figure 2: Diagrama casos de uso usuario

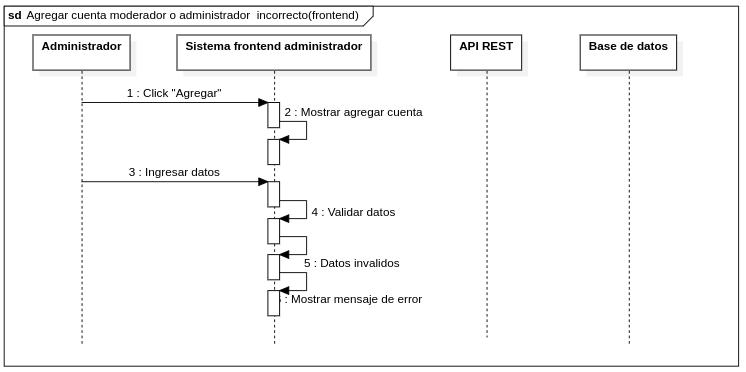
|  |
| --- |
| **Diagrama de secuencia** |

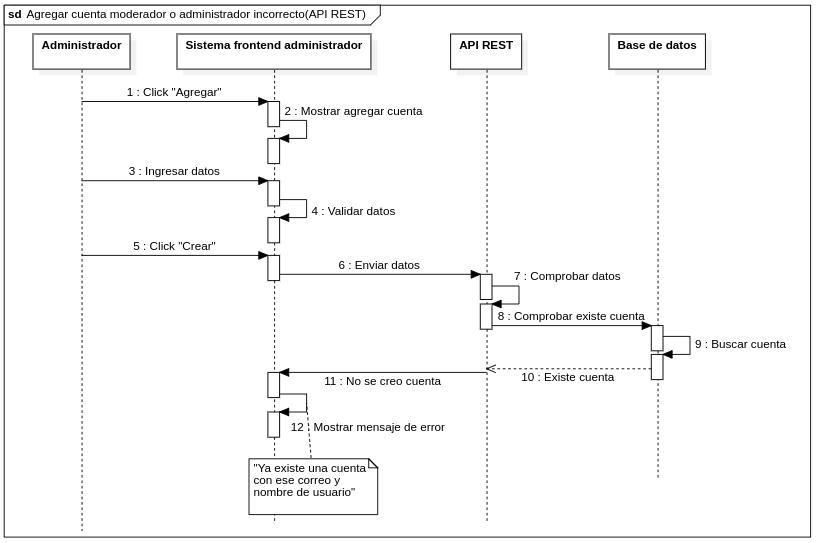
Figure 3: Inicio de sesión correcto

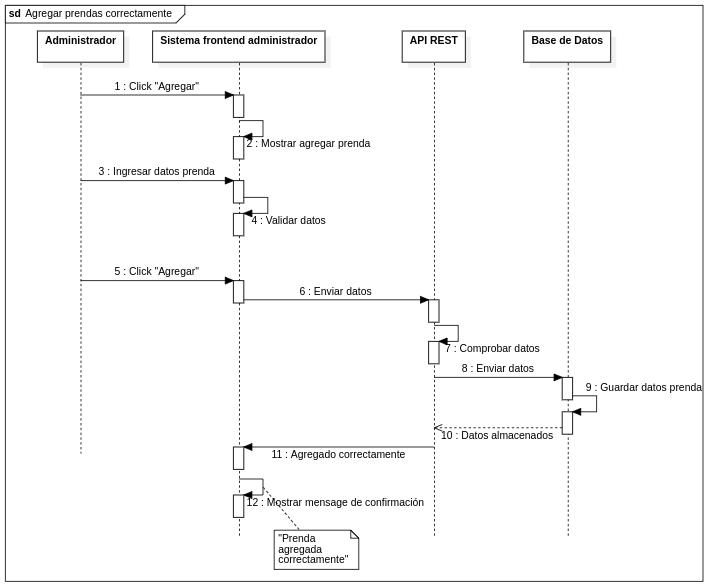
Figure 2: Inicio sesión incorrecto(no encontrado admin-moderador)

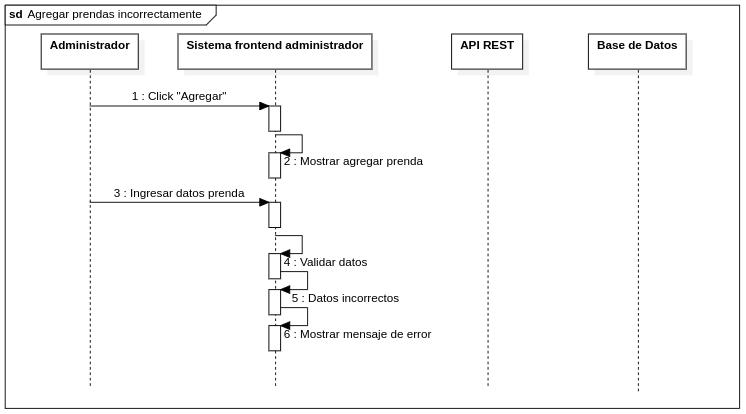
Figure 4: Inicio sesión incorrecto(clave incorrecta)

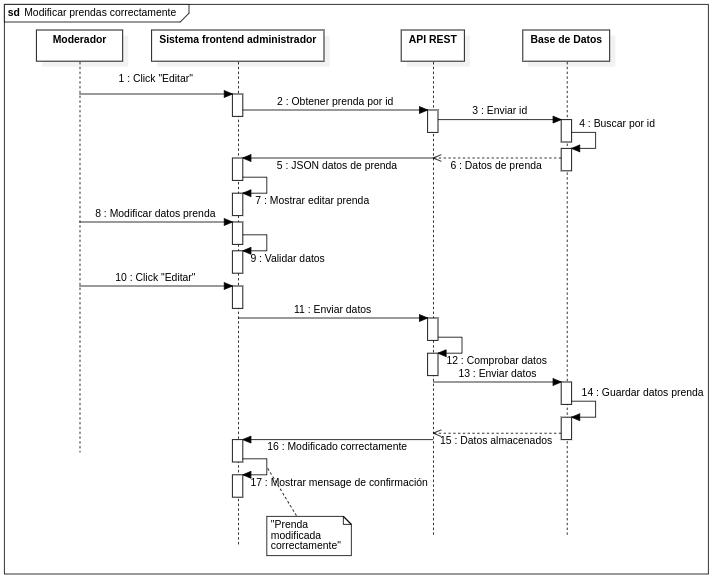
Figure 4: Agregar cuenta moderador o administrador

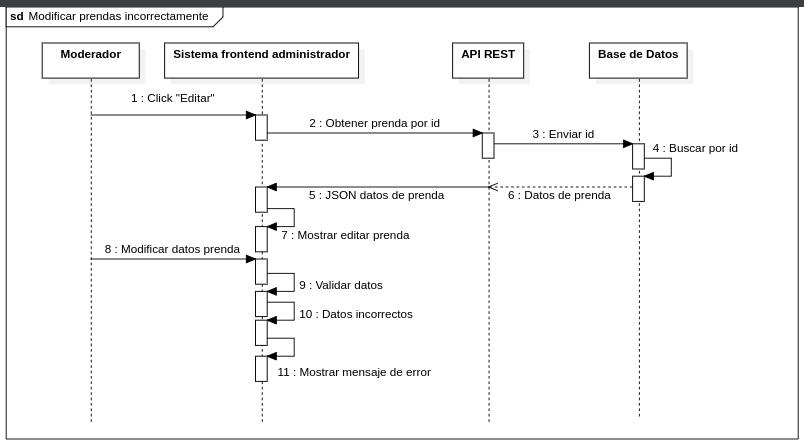
Figure 5: Agregar cuenta moderador o administrador incorrecto(frontend)

Figure 6: Agregar cuenta moderador o administrador incorrecto(API REST)

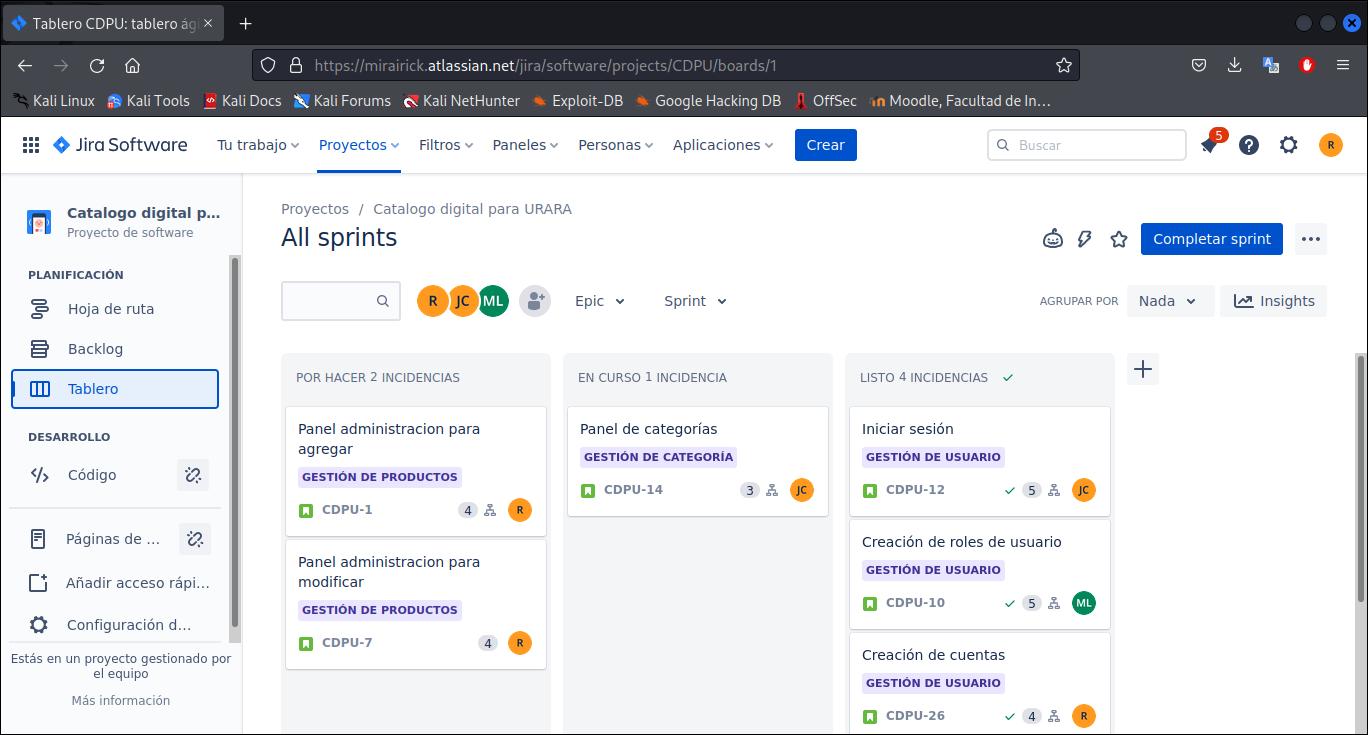
Figure 6: Agregar prendas correctamente

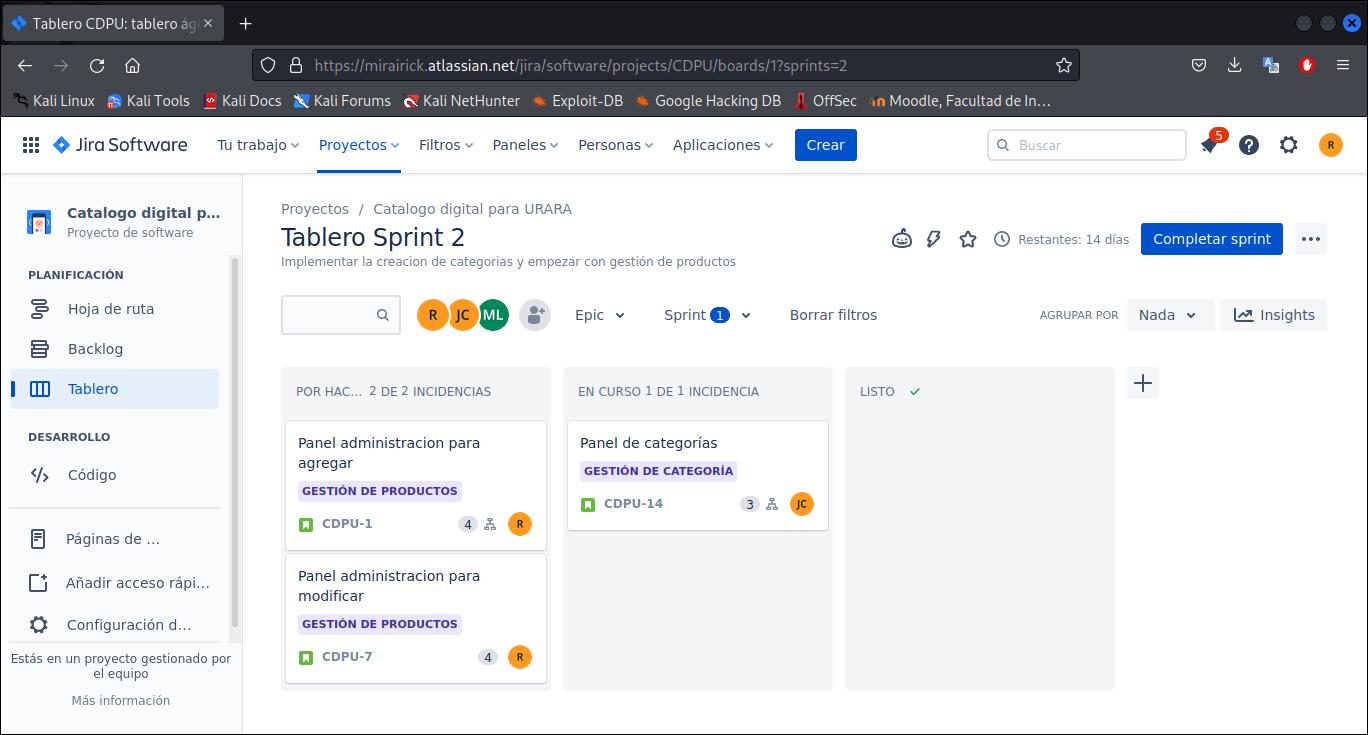
Figure 7: Agregar prendas incorrectamente

Figure 8: Modificar prendas correctamente

Figure 9: Modificar prendas incorrectamente

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

Figure 10: Todos los sprints

Figure 11: Segundo sprint