UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Ingeniería de Software y Sistemas Informáticos

Sistema para la gestión de proyectos de micropropagación de plantas biotecnológicas.

Trabajo Fin de Máster
Tipo de trabajo: Desarrollo práctico
Presentado por: Reyes Bermúdez, Enrique
Director/a: Diaz Piraquive, Flor Nancy

Ciudad: Quito
Fecha: septiembre del 2018

Resumen

El cultivo de tejidos vegetales, como método de propagación, es una de las técnicas más utilizada para la producción de plantas que son difíciles de propagar a través de los métodos convencionales. La producción de plantas *in vitro* en Laboratorios Comerciales, exige de procesos y operaciones de trabajo estrechamente relacionadas entre sí, entre ellos, los métodos de planificación de la producción para definir con anticipación los requerimientos humanos y materiales para su realización. La planificación puede realizarse mediante el uso de un software apropiado y puede ser organizado utilizando un sistema en red que permita el enlace de todas las áreas. En el presente trabajo se utilizaron elementos y variables, considerados indispensables, para la planificación de la producción. El desarrollo del producto se realizó usando la metodología de trabajo RUP, que permite transformar los requisitos de usuario en un sistema o software a través de su modelo de implementación en espiral, así como sus artefactos generados y modelados a través del UML. El aplicativo tiene una arquitectura cliente-servidor, donde el cliente (frontend) consume el servicio que es brindado por el servidor (backend), implementando cada uno por separado la arquitectura MVC con sus diferentes patrones de diseño. En el backend se utilizó Java como lenguaje de programación, Spring como framework de desarrollo de aplicaciones web, Apache Tomcat como servidor de aplicaciones y PostgreSQL como gestor de base de datos. El frontend se desarrolló utilizando el framework Angular y Typescript como lenguaje de programación encargado de generar el código JavaScript interpretado por el navegador web. Se utilizaron además el IDE Eclipse y el editor de código Visual Studio Code.

Palabras clave: Backend, Biofábricas, Frontend, In Vitro, Micropropagación.

Abstract

Plant tissue culture, as a method for plant propagation, is one of the most widely used techniques for the production of plants that are difficult to propagate through conventional methods. The production of *in vitro* plants in Commercial Laboratories requires working processes and operations closely related to each other, among them, planning methods to define in advance both the human and material resources necessaries. Planning can be done through the use of appropriate software and can be organized using a network system that allows the linking of areas. In this work, some elements and variables, considered indispensable, were used for production planning. Product development was made using the RUP work methodology, which allows to transform user requirements into a software system through its spiral implementation model, as well as its artifacts generated and modeled through the UML. The application had a client-server architecture, where the client (frontend) consumes the service that is provided by the server (backend), implementing each one separately the MVC architecture with its different design patterns. In the backend Java was used as a programming language, Spring as a web application development framework, Apache Tomcat as application server and PostgreSQL as a database manager. The frontend was developed using the Angular and Typescript framework as a programming language responsible for generating the JavaScript code interpreted by the web browser. The Eclipse IDE and the Visual Studio Code editor were also used.

Keywords: Backend, Biofactories, Frontend, In Vitro, Micropropagation

Índice de tablas

Índice de figuras

1. Introducción

1.1. Justificación

El sector agrícola busca constantemente técnicas innovadoras y modernas para aumentar la eficiencia de los cultivos y producir alimentos seguros, piensos y fibras de alta calidad a precios asequibles. La micropropagación por cultivo de tejidos se utiliza cada vez más para reproducir cultivos que son difíciles de propagar por métodos convencionales. Se ha demostrado que este es un método de propagación altamente eficiente para la producción de grandes cantidades de plantas idénticas y libres de enfermedades en un espacio reducido y menor cantidad de material a propagar (Rangel-Estrada, Hernández-Meneses, & Hernández-Arenas, 2016).

La micropropagación por cultivo de tejidos es una colección de técnicas usadas para mantener o cultivar células, tejidos u órganos vegetales en condiciones estériles, en un medio de cultivo nutritivo de composición conocida. El éxito en la producción de grandes cantidades de plántulas requiere de instalaciones estéril, un medio de cultivo ideal y expertos con experiencia para supervisar estrictamente los protocolos (Kadam, Chhatre, Lavale, & Shinde, 2018).

Las biofábricas se pueden definir como un centro para la producción a gran escala de plantas y semillas mejoradas que permite obtener ejemplares con novedosas técnicas científicas que garantizan su calidad. Estas se constituyen en una innovación tecnológica respecto a los sistemas tradicionales de producción de semilla. Se han desarrollado de forma continua en el tiempo, convirtiéndolo en un modelo muy competitivo a nivel internacional. El análisis comparativo de variables entre los sistemas tradicionales de propagación y la producción en biofábricas, confirma las ventajas que poseen estas últimas y su valor estratégico (Agramonte, y otros, 2006).

El costo de la mano de obra, aproximadamente el 70% del costo total de estos procesos tecnológicos, se puede reducir mejorando los protocolos de micropropagación, aumentando la automatización tanto de la preparación del medio como del lavado de la cristalería, así como con la automatización de la fase de multiplicación, con los denominados biorreactores de inmersión temporal (Galvão Mendonça, y otros, 2016).

Se ha demostrado que la producción de vitroplantas a gran escala en biofábrica es un proceso rentable, por ejemplo, el banano como principal renglón de exportación agrícola en Ecuador, al ser propagado por cultivo de tejidos resultó más rentable que el banano propagado convencionalmente (Alagumani, 2005). Estas tecnologías también han sido usadas eficientemente para la producción de otras especies de plantas, que incluye

adicionalmente a la propagación masiva de plantas, las tecnologías de micropropagación también han sido pilares fundamentales en la producción a gran escala de metabolitos secundarios vegetales con amplio uso en la industria biotecnológica, medico-farmacéutica y alimenticia (Bhatia & Bera, 2015). Sin embargo, la implementación de sistemas informáticos para el aseguramiento de la calidad de las producciones y el análisis de sus costos es determinante en el éxito de los procesos de producción en biofábricas.

1.2. Planteamiento del problema

Las tecnologías de micropropagación, con el empleo de protocolos para cultivo de células y tejidos vegetales han devenido en procesos tecnológicos viables en la producción de propágulos con rasgos genéticos deseados y excelente calidad fitosanitaria. La propagación de vitroplantas en laboratorios comerciales se sustenta en procedimientos operacionales estrictamente relacionados. De ellos, la estricta planificación del proceso productivo en función de las demandas del mercado es decisivo para el éxito comercial, así como la definición de las necesidades y costos de producción.

No existe un software disponible que soporte el desarrollo tecnológico de los laboratorios actuales. El presente trabajo se sustenta en el desarrollo actual de la micropropagación de plantas por vía biotecnológica como proceso tecnológico y la definición previa de elementos y variables, directamente relacionados con el flujo productivo y la eficiencia del proceso, que permita obtener los menores costos unitarios para la comercialización de las vitroplantas con los mayores estándares de calidad.

1.3. Estructura de la memoria

La presente memoria está constituida por los siguientes capítulos:

Capítulo 1: Se realiza una introducción al trabajo presentado, a través de la justificación del tema que se abordará y el planteamiento del problema.

Capítulo 2: Se explica el estado del arte a través del concepto de micropropagación y sus antecedentes.

Capítulo 3: Se definen los objetivos generales y específicos. Se detalla la metodología de trabajo.

Capítulo 4: Se abordan los conceptos principales sobre las herramientas y tecnologías utilizadas para el desarrollo de la solución propuesta.

Capítulo 5: Se describe la solución propuesta a través de los requerimientos funcionales y no funcionales, el diagrama de casos de uso, los diagramas de clases, los diagramas de componentes y el diagrama de despliegue.

Capítulo 6: Se presentan las conclusiones generales y recomendaciones para la continuidad del trabajo.

Capítulo 7: Se brindan la bibliografía utilizada que sirve como sustento a la investigación científica realizada.

Capítulo 8: Se anexan documentos adicionales utilizados.

Capítulo 9: Se ofrece la información técnica adicional sobre el tema desarrollado.

2. Contexto y estado del arte

La biotecnología vegetal se ha desarrollado vertiginosamente desde las últimas décadas del pasado siglo, fundamentado por una parte en la obtención de plantas con caracteres mejorados a partir de la introgresión dirigida de genes deseables y, por otra, la introducción de nuevas las variedades mejoradas a la práctica agrícola, tolerantes o resistentes a factores bióticos o abióticos. Esto requiere de protocolos eficientes de propagación de plantas a gran escala.

La aplicación de la micropropagación data del inicio de los años 70 del siglo XX, mayoritariamente en investigaciones con especies hortícolas. Sin embargo, cientos de tecnologías han sido desarrolladas para explotar el potencial genético de las plantas cultivadas, así como la variabilidad de plantas ornamentales.

Los costos de producción han conllevado al desplazamiento de los laboratorios comerciales inicialmente desarrollados en los llamados países del “primer mundo”, con un alto costo de mano de obra, hacia países menos desarrollados. Otra estrategia ha sido la automatización de los procesos productivos, con el empleo de los denominados biorreactores de inmersión temporal o los protocolos de embriogénesis somática, usando fermentadores adaptados a partir de los desarrollados para la producción de vacunas.

Obviamente, el avance de estas tecnologías ha sido el producto de alianzas estratégicas entre universidades o grupos de investigación y entidades productivas. No obstante, la optimización de estos procesos productivos requiere del desarrollo de sistemas informáticos que permitan modelar los componentes del proceso tecnológico y los costos incurridos en las diferentes etapas, como se describen próximamente en el proceso de micropropagación.

Lo anterior permitiría adecuar las capacidades productivas a las demandas del mercado y, por ende, garantizar los compromisos contractuales en relación a volumen, calidad y momentos de entrega. Además, garantiza el cálculo oportuno de los costos de producción y la definición de las utilidades necesarias para el crecimiento empresarial.

Hasta nuestro nivel de conocimiento, los aspectos anteriores se tratan aisladamente. En el presente trabajo experimental hemos integrado las diferentes etapas del proceso de micropropagación, a fin de ofrecer un software que permita asegurar el flujo productivo, en relación a contratos con clientes, las capacidades productivas, los recursos materiales y humanos, la calidad del proceso y los costos de producción incurridos. Ello permitirá establecer precios competitivos y volúmenes de plantas de calidad.

2.1. Micropropagación

La micropropagación de plantas se ha comercializado para producir plántulas de alta calidad y libres de virus. Alto costo y trabajos intensos son los principales problemas con esta técnica (Orellana, y otros, 2008).

Los gastos directos de micropropagación se pueden dividir en dos grupos, que incluyen gastos de mano de obra, materiales y electricidad, y la depreciación de la infraestructura. Los gastos directamente relacionados con la realización de una tecnología de propagación se definen como costos de operación. Estos podrían fácilmente cambiarse y están bajo el control constante del gerente de laboratorio (Hempel, Хемпел, & Хемпел, 1986).

Los costos de operación pueden dividirse en el trasplante de las plantas, la preparación de los medios y la limpieza de los recipientes de vidrio y de cultivo. Se distribuyen de acuerdo con la participación que tienen en el costo total de mano de obra.

Componentes del costo variable en la micropropagación:

1. Costos laborales (mano de obra).

1.1. Trasplante del material vegetal

Es evidente que la estrategia para la reducción de costos debe concentrarse en la reducción del costo del trabajo en la cabina de flujo de laminar. Esto se puede lograr de dos maneras: mejora de la tecnología de propagación y una mejor organización del trabajo.

La mayoría de los métodos de propagación *in vitro* constan de tres etapas. En la primera etapa (fase I), las puntas de los brotes o partes de otros órganos se aíslan, los explantes se ponen en nuevas condiciones de cultivo y se induce la organogénesis de los brotes (obtención de nuevos brotes). Los brotes obtenidos se multiplican para la obtención de nuevos explantes (fase II) y los brotes resultantes (luego de separados nuevamente) se enraízan, pasando los mismos a un nuevo frasco de cultivo con auxinas en el medio de cultivo, antes de plantar en el invernadero (fase III).

La eficiencia del trabajo de corte de los brotes (fase II, multiplicación) puede mejorar por un aumento del coeficiente de multiplicación de cada brote implantado previamente. De este modo se reduce el tiempo necesario para la apertura y cierre de los recipientes de cultivo y para extraer los brotes multiplicados y plantar los brotes individuales.

Por otra parte, con un mayor coeficiente de multiplicación se logra un mayor volumen de producción en menos tiempo y se reduce el costo unitario de las vitroplantas. La

eficiencia de los trabajadores puede mejorarse con el uso de herramientas apropiadas y equipos de esterilización.

1.2. Preparación de medios

La segunda parte de los costos laborales es el gasto en la preparación de los medios de cultivos. La reducción de estos costos puede lograrse mediante el uso de medios preparados o soluciones madres que reducen el tiempo necesario para pesar y disolver los ingredientes de los medios individuales.

1.3. Limpieza de los recipientes de cultivo y de la cristalería del laboratorio.

El gasto de dinero en el trabajo necesario para lavar los recipientes y la limpieza del laboratorio tienen una participación significativa en el presupuesto del laboratorio. Se puede reducir utilizando lavadoras de vidrio, que no solo limpian el vidrio más rápido, sino que también lo esterilizan parcialmente.

2. Materiales

La segunda parte de los costos operacionales son gastos en materiales, tales como componentes de medios, recipientes de cultivo y material vegetal.

2.1. Ingredientes de los medios

Disminuir el costo de los medios significa la reducción del volumen de medios necesarios para la producción de un nuevo explante (callos, brotes, brotes enraizados, etc.). Esto podría lograrse mediante la mejora del coeficiente de multiplicación o el enraizamiento de los brotes. Se puede obtener un efecto similar al de reducir la cantidad de medio por cada explante plantado en cada recipiente de cultivo.

El gasto en los medios también se puede disminuir por la reducción del contenido o la eliminación de los componentes más caros de los medios, por ejemplo, el Agar. Ese es el ingrediente más caro de los medios de cultivos. Es bien conocido que puede eliminarse de un medio sin retrasar el crecimiento de la planta, pero habría que ajustar las tecnologías del cultivo *in vitro* para especies de cultivos determinadas. Para este fin se han desarrollado los denominados biorreactores de inmersión temporal, los cuales también reducen el costo de la mano de obra; aunque no han sido ampliamente difundidos en las biofábricas.

2.2. Recipientes de cultivo

La depreciación de los recipientes de cultivo también se incluye en los costos de los materiales, ya que en la mayoría de los laboratorios no se usan más de unos pocos años. El costo de los recipientes puede disminuir si se mejora el coeficiente de

multiplicación y el porcentaje de enraizamiento, si se siembran más explantes en un contenedor de cultivo (si el cultivo lo permitiere) o el uso de recipientes más baratos o duraderos.

Las dos primeras estrategias son similares para el caso del costo de los medios de cultivos. En la práctica, los problemas de la densidad de plantas y el volumen de los medios en un recipiente deben considerarse mutuamente. A mayor número de brotes, mayor número de plantas cultivadas. Sin embargo, ello puede afectar la calidad de las plantas y su estadía en la fase de viveros (fase IV).

2.3. Material vegetal

La tercera parte, y la menos importante, del costo del material es el gasto en el material a partir del cual los explantes serán aislados al comienzo del esquema de propagación. Estos costos dependen directamente del tiempo de multiplicación *in vitro* de estos explantes. En la medida que se produzcan más plantas a partir de un explante inicial y en menor tiempo, menor será el costo del material vegetal por planta de una progenie (clon).

La tasa de contaminación de los cultivos iniciales también tiene un efecto directo sobre la participación del costo del material vegetal en el gasto material total. El uso de explantes aislados de una planta madre sin destruirla o disminuir sus cualidades de producción puede verse como una forma de reducir estos costos.

Otros gastos en los que se incurren y en ocasiones no están directamente relacionados con los niveles de producción lo constituye la electricidad; mayoritariamente relacionado al mantenimiento de la climatización de los locales por el uso de Split o aires acondiciones y la iluminación. Este componente conjuntamente con la depreciación de la infraestructura y equipamiento, frecuentemente se analizan como un coeficiente de gastos indirectos sobre los costos de producción (Sahu & Kumar Sahu, 2013).

Según (Chen, 2016), en los componentes del costo incluyen mano de obra, supervisión, ventas y mercadeo, infraestructura y equipamiento, energía, agua y materias primas. El mayor costo lo constituye la mano de obra calificada para la transferencia de las vitroplantas. Los factores que afectan el costo unitario incluyen la tasa de pérdidas por contaminación o mal manejo de los protocolos de micropropagación, la capacidad de producción, el coeficiente de multiplicación y el salario diario. El coeficiente de multiplicación se destaca como el factor principal en el costo unitario, así como en el período total requerido.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

El siguiente trabajo se plantea como objetivo general desarrollar una aplicación web para la gestión de proyectos de micropropagación a gran escala de plantas biotecnológicas y la generación de costos de producción bajo los principios del software libre.

3.2. Objetivos específicos

- Definir los requerimientos necesarios que debe cumplir el sistema para que sea totalmente operativo.
- Desarrollar la aplicación que responda a los requerimientos definidos y generar la documentación correspondiente, que servirá como apoyo a las implementaciones y mejoras futuras.
- Realizar las pruebas a software desarrollado con el objetivo de validar sus funcionalidades.
- Evaluar el aplicativo a través de la opinión de especialistas en el negocio.

3.3. Metodología de trabajo: RUP

El proceso de desarrollo de software se define como “el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software” (Jacobson, Booch, & Rumbaugh, 2000, pág. 4).

Según (UNIR, Metodologías, Desarrollo y Calidad de la Ingeniería de Software (ISW) - PER5 2017-2018 - Tema 2, 2018), el proceso unificado se apoya en UML para modelar todos los artefactos del software, definiendo los siguientes aspectos:

- ✓ Dirigido por casos de uso: Los casos de uso representan el inicio de un conjunto de actividades que contribuyen al proceso de desarrollo del software.
- ✓ Centrado en la arquitectura: La arquitectura de un software se representa a través de distintas vistas estáticas y dinámicas, ofreciéndole al desarrollador varias perspectivas del producto que implementará.
- ✓ Iterativo e incremental: Tiene como principio dividir el trabajo en unidades más pequeñas, donde cada actividad se ejecuta de forma iterativa y el crecimiento del producto representa el incremento.

El Proceso Unificado reincide a lo largo de un conjunto de ciclos que representan la vida de un sistema y todo ciclo culmina con un entregable a los clientes. Cada ciclo se ejecuta a lo largo del tiempo, dividiéndose en cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase culmina con un hito que representa la existencia de una serie de artefactos, es decir, modelos y documentos que han sido desarrollados. Dentro de las fases se puede descomponer el trabajo en varias iteraciones, y una iteración atraviesa por los siguientes flujos de trabajo: Requisitos, Análisis, Diseño, Implementación y Pruebas (Jacobson, Booch, & Rumbaugh, 2000, págs. 9-11).

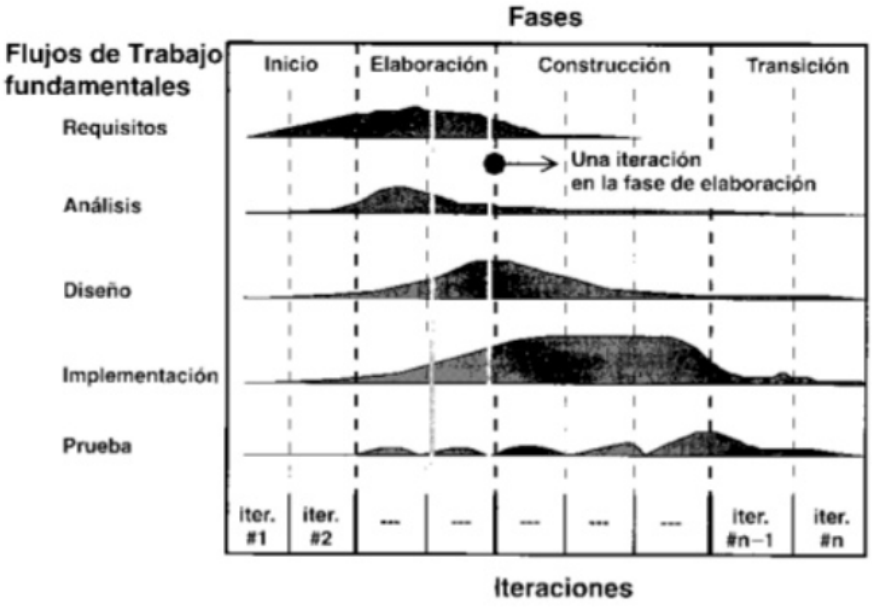


Ilustración 1: Fases y flujos de trabajo del Proceso Unificado

4. Herramientas y tecnologías utilizadas

4.1. Lenguaje de modelado: UML

Según (Rumbaugh, Jacobson, & Booch, 2000, pág. XVII), UML representa mucho más que un lenguaje de programación. En sus orígenes se parece a C++ o a Java. Si se le compara con lenguajes claves del mundo de la Internet como HTML, Java y XML, se observa que UML se ha diseñado y construido como un lenguaje de una madurez acentuada.

"El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software" (Larman, 2003, pág. 10). (Fowler & Scott, 1997, pág. 1) explica que el UML no es un método, sino un lenguaje de modelado. Una vista por su parte, es un área dentro de UML, que describe un determinado punto de vista del sistema que se está desarrollando.

Para obtener una mejor visión de la solución desarrollada se modelará el sistema a través de los siguientes diagramas:

- Casos de uso.
- Clases.
- Secuencia
- Paquetes.
- Despliegue.

4.2. Lenguajes de programación

Para escribir un programa para ordenadores es necesario utilizar un lenguaje de programación. Los lenguajes de programación están compuestos por una serie de palabras claves cumpliendo ciertas reglas sintácticas (Solano, 2011).

Un lenguaje de programación se define como "un idioma artificial diseñado para que sea fácilmente entendible por un humano e interpretable por una máquina" (Jiménez Marín & Pérez Montes, 2016, pág. 3).

Existen gran variedad de lenguajes de programación, cada uno con sus ventajas y desventajas, siempre teniendo en cuenta el entorno donde será utilizado. Actualmente los lenguajes de programación permiten escribir código legible y sin muchas dificultades, utilizando un compilador para generar el código binario que es entendido por los

ordenadores. Java es un lenguaje de propósito general, encontrándose entre los primeros escaños como uno de los lenguajes más utilizados (Jiménez Marín & Pérez Montes, 2016, págs. 3,4).

4.2.1. Lenguajes de programación disponibles en el backend

Para el desarrollo del backend existen diversos lenguajes de programación, cada uno con sus características, ventajas y desventajas. A continuación, se realiza una comparación entre tres de los lenguajes más utilizados en la actualidad (Tabla 1).

Tabla 1: Comparación de lenguajes de programación

	.NET	JAVA	PHP
Frameworks y tecnologías de desarrollo web	MVC, WebAPI, WCF.	Spring, JSF, Struts, Play, entre otros.	Laravel, Symfony, Zend, Code Igniter, Yii, entre otros.
Rendimiento y velocidad	Muy rápido, compila a código nativo.	Menos rápido que .NET, la máquina virtual interpreta el código intermedio.	Menos rápido que .NET, el servidor interpreta el código PHP.
Tipado	Fuertemente tipado.	Fuertemente tipado.	Débilmente tipado.
Curva de aprendizaje	Media	Alta	Baja
Despliegue	Se debe compilar el código ante cualquier cambio.	Se debe compilar el código ante cualquier cambio.	Se sustituye el código desplegado.
Editores y herramientas	Visual Studio.	Eclipse, NetBeans.	Aptana Studio, Eclipse, NetBeans, Visual Studio Code, Notepad++.
Plataforma	Windows, Linux (NET Core)	Windows, Linux, Mac OS	Windows, Linux, Mac OS
Sitios populares desarrollados	MSN, Live.com	Google, Youtube, Yahoo, Amazon	Google, Facebook, Yahoo, Wikipedia, WordPress

Para la construcción de la aplicación es necesario un lenguaje de programación que permita el desarrollo bajo los principios del software libre y que fuese fuertemente tipado, con el objetivo de disminuir errores debido a la interpretación del tipo de variable en diferentes ámbitos. Teniendo en cuenta la comparación anterior, se seleccionó a Java como el lenguaje de programación que se empleará en el desarrollo del backend.

4.2.2. Java

El lenguaje Java fue creado para ser utilizado en el mundo de los dispositivos electrónicos. Debido a la enorme variedad de equipos electrodomésticos, era necesario encontrar un mecanismo que independizara la especificación del software del hardware donde se utilice, es decir, una capa intermedia. En el año 1995 Java es utilizado como lenguaje de programación para computadoras, tras la fallida aceptación inicial de las empresas de electrodomésticos (Torres Remon, 2013, pág. 13).

Según (Sánchez Asenjo, 2009, pág. 11), la sintaxis del lenguaje Java es muy parecida a C y C++, aunque se evidencian las siguientes diferencias:

- ✓ No existen punteros, aumentando su seguridad y facilidad de uso.
- ✓ Es completamente orientado a objetos.
- ✓ Preparado para ser utilizado en aplicaciones de redes TCP/IP.
- ✓ Las excepciones son implementadas de forma nativa.
- ✓ Es un lenguaje interpretado, facilitando su ejecución remota y disminuyendo su rapidez en comparación con las aplicaciones escritas en lenguajes compilados como C++.
- ✓ Es multihilo, facilitando la ejecución de tareas en paralelo.
- ✓ Fuertemente tipado.
- ✓ Independiente de la plataforma, se ejecuta sobre una máquina virtual.

Java es utilizado en multitud de ámbitos y tecnologías debido a sus características de seguridad, portabilidad, recurrencia, etc. En dependencia del ámbito donde se vaya a trabajar se pueden utilizar distintas plataformas: Java Standard Edition, Java Enterprise Edition, Java Card y Java Micro Edition. La plataforma Java EE implementa las APIs y funcionalidades necesarias para poder ejecutar aplicaciones servidoras (Ordax Cassá & Ocaña Díaz-Ufano).

En el backend de la solución desarrollada se utilizará la plataforma Java EE para la construcción de un servicio que podrá ser consumido por cualquier cliente independientemente de la tecnología, lenguaje de programación, arquitectura, bibliotecas y framework seleccionados. La comunicación se podrá realizar siempre que se realicen correctamente las llamadas a las interfaces definidas por el servicio.

4.2.2.1. Apache Tomcat

Apache Tomcat es un servidor open source, contenedor de aplicaciones web, que se creó para ejecutar aplicaciones web servlet y Java Server Pages (JSP). Inicialmente fue creado por el proyecto Apache-Jakarta, y actualmente se encuentra alojado como un proyecto de Apache debido a su popularidad, mantenido además por la comunidad de código abierto de Java. Se encuentra bajo la licencia Open Source Apache, caracterizándose por su estabilidad y demás características de un contenedor de aplicaciones web comercial (Vukotic & Goodwill, 2011, pág. 1).

El software Apache Tomcat potencia numerosas aplicaciones web de alta criticidad a gran escala en innumerables organizaciones. Algunos de estos clientes se enumeran en la página wiki de PoweredBy (Foundation, s.f.).

4.2.3. JavaScript

Debido a la complejidad de aplicaciones web iniciales y a la escasa velocidad de navegación, surge la necesidad de crea un lenguaje que se ejecutara del lado del navegador, con el objetivo de realizar validaciones y disminuir el tráfico de red. Inicialmente Netscape creó a LiveScript y posteriormente del fruto de su alianza con Sun Microsystems, surge JavaScript. Su nombre se debe al enorme significado que representaba Java en esos momentos, es decir, puramente por marketing. La primera versión de JavaScript fue un éxito, lanzándose JScript por parte de Microsoft al mismo tiempo, cuya implementación era la misma que JavaScript, pero con nombre diferente para evitar problemas legales. Luego se decidió estandarizar a JavaScript por el organismo ECMA[1], obteniéndose como resultado la definición del lenguaje ECMAScript. La denominación ECMAScript es de preferencia por algunos programadores (Eguíluz Pérez, 2009, págs. 5,6).

JavaScript ofrece una mayor interactividad con el usuario y posee interfaces enriquecidas mediante controles avanzados. Entre sus limitaciones se encuentran la

[1] European Computer Manufactures Association

imposibilidad de leer o escribir archivos en el sistema operativo donde se ejecute, funcionalidad que no se le ha agregado actualmente por razones de seguridad (Javascript language, 2015, págs. 11,12).

Es necesario aclarar que JavaScript y Java son dos lenguajes de programación diferentes, a pesar de tener nombres similares. Mediante un editor de texto plano se puede programar en JavaScript, sin la necesidad de herramientas o editores avanzados (Mohedano, Saiz, & Salazar Román, 2012, págs. 9,10).

4.2.3.1. TypeScript

TypeScript es un lenguaje de programación open source desarrollado por Microsoft. El participante principal del desarrollo del lenguaje es Anders Hejlsberg, siendo además el arquitecto principal del desarrollo del lenguaje de programación C#. El lenguaje es llamado como Superset de JavaScript, de tal forma que el navegador, encargado de interpretar el código JavaScript, nunca sabrán que el código original fue TypeScript (que-es-typescript, s.f.).

TypeScript es considerado como un conjunto de herramientas que facilita el desarrollo de aplicaciones más potentes y utiliza los mismos avances añadidos por ECMAScript, permitiéndole a los desarrolladores adelantarse al futuro utilizando las mejoras del lenguaje (typescript-javascript, s.f.).

Todo programa en JavaScript se puede generar a partir de un programa en TypeScript, sin la necesidad de transformar el nombre de las variables declaradas, permitiendo de esa manera su depuración directa. La sintaxis de TypeScript incluye varias características propuestas por ES6[2], entre las que se encuentras las clases y los módulos, facilitando el uso de la orientación a objetos y de la organización de los componentes y módulos para evitar conflictos (TypeScript, 2013).

El lenguaje es utilizado en múltiples frameworks de desarrollo, facilitando su implementación y mantenimiento. En la solución desarrollada se utiliza TypeScript para el desarrollo de frontend.

4.3. Frameworks

(Gutiérrez) nos define a un framework como “una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.”

[2] ECMAScript 6

Framework es un concepto que se utiliza en múltiples esperas dentro del desarrollo de un software. Se pueden utilizar diferentes frameworks para el desarrollo de aplicaciones web, videojuegos, visión por computador, y para infinidades de entornos. Mediante los framework se puede reutilizar código y acelerar el proceso de desarrollo. En su mayoría, los frameworks web implementan el patrón Modelo - Vista - Controlador, fundamental para definir arquitecturas que ofrecen una fuerte interacción con el usuario. Entre las características fundamentales de los framework web se encuentran su facilidad de integración con las bases de datos, la abstracción de las URL, sus mecanismos de autenticación y control de acceso, la internacionalización y la separación del diseño y el contenido (Gutiérrez, págs. 1-4).

El uso de un framework también tiene sus limitaciones, entre las que se pueden mencionar su elevado tiempo de aprendizaje, el exceso de código y su código público, permitiendo este último a un hacker su estudio detallado para realizar un ataque teniendo en cuenta las debilidades encontradas (Framework para el desarrollo ágil de aplicaciones).

Los framework están formados por zonas congeladas y zonas calientes. Las zonas congeladas permanecen inalterables en cualquier instanciación del framework, es decir, definen la arquitectura. Las zonas calientes les permiten a los programadores agregar su código respecto a las funcionalidades que desea utilizar. Existen los frameworks de caja blanca, que exigen el conocimiento del usuario sobre su estructura y código y en su mayoría brindan su código fuente; y los frameworks de caja negra, que no necesitan que el usuario conozca su funcionamiento interno (Gutierrez, 2010, págs. 12,15).

4.3.1. Spring Framework

Spring Framework es uno de los framework open source destacados para el desarrollo de aplicaciones dentro de la plataforma de desarrollo J2EE, permitiendo el desarrollo eficiente desde aplicaciones sencillas hasta las más complejas. Entre los principales conceptos que Spring Framework depende podemos encontrar la inyección de dependencia (DI), la programación orientada a aspectos (AOP) y la API de persistencia de Java (JPA). Spring framework está formado por casi 20 módulos, ver Ilustración 2. Entre sus módulos se encuentran Core Container, Data Access/Integration y Web. El módulo Web es el más importante para el desarrollo de aplicaciones web, integrado por los módulos Web, WebSockets, Portlet y Servlet. El módulo Servlet contiene dos de los conceptos más utilizados actualmente: Spring Model-View-Controller (MVC) y RESTful WebSerive (REST WS). Mediante Spring MVC se puede desarrollar aplicaciones en

varias capas, permitiendo su expansión y denegando su modificación. REST WS permite la interoperabilidad entre sistemas informáticos, convirtiéndose hoy día en uno de los mecanismos más utilizadas para proveer datos a diversos tipos de consumidores (Jovanović, Jagodić, Vujičić, & Ranđić, 2017).

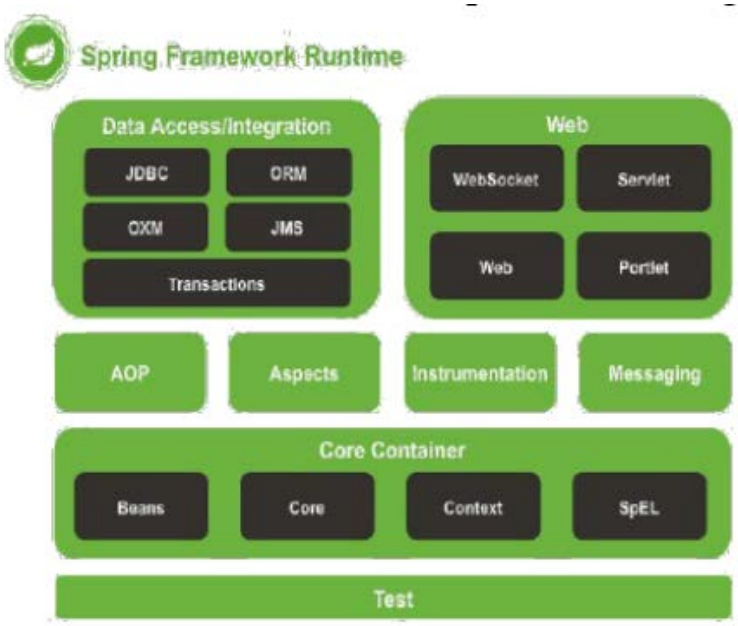


Ilustración 2: Arquitectura de Spring Framework

4.3.1.1. Spring Boot

El objetivo de Spring Boot es facilitar el desarrollo de aplicaciones Spring (Spring Framework), a través de la configuración automática, la configuración de las aplicaciones como Standard Spring Applications, la automatización de las dependencias requeridas, el control de la aplicación a través de la consola, entre otras funcionalidades (Jovanović, Jagodić, Vujičić, & Ranđić, 2017).

Spring Boot provee los mecanismos para el desarrollo de microservicios, permitiendo además el empaquetado de sus programas para que sean ejecutados como una aplicación alojada en un servidor de aplicaciones, o ejecutarse como un archivo .jar ejecutable que contiene integradas sus dependencias y algún tipo de servidor embebido, usualmente Apache **Tomcat** (Hofmann, Schnabel, & Stanley, 2016, pág. 10).

4.3.2. Angular

Angular es un framework JavaScript del lado cliente desarrollado por Google. La versión inicial de Angular se nombró AngularJS, luego se rescribió el framework completamente

bajo el nombre de Angular 2. La versión actual de Angular es la 6, pero se refiere a la arquitectura de Angular 2, es decir, AngularJS es un producto distinto a Angular 2.

A través de una arquitectura MVC, Angular 2 facilita la implementación de aplicaciones web bien diseñada y bien estructuradas, proporcionando todas las funcionalidades que permitan manejar las entradas del usuario en el navegador, manipular los datos en el cliente y controlar cómo se muestran los elementos en el navegador.

Entre sus principales beneficios podemos mencionar:

- ✓ Facilidad de vincular datos a elementos HTML a través del “Data binding”.
- ✓ Extensibilidad, permitiéndole al usuario personalizar su propia implementación.
- ✓ Estricta obligación al usuario para escribir código limpio y lógico
- ✓ Flexibilidad para escribir código reutilizable.
- ✓ Soporte a través de Google como anfitrión.

Compatibilidad al establecer una estrecha relación con el estándar de JavaScript.

(Dayley, Dayley, & Dayley, 2018)

En la solución propuesta se utiliza a Angular 2 como framework de desarrollo en el frontend, con el apoyo de los diferentes paquetes que facilitan la reutilización de código.

4.4. Servicios Web REST

Los servicios web están formados por APIs que pueden ser invocadas y que son ejecutadas dentro del propio sistema donde se definen. En los últimos tiempos ha surgido un nuevo estilo arquitectónico de software llamado REST (Representational State Transfer), proporcionando una nueva alternativa para el desarrollo de los servicios web (Navarro Marset, 2007, pág. 3).

Dentro de las principales características de REST frente a SOAP, se destaca la ausencia de estados. Quiere decir que diferentes llamadas realizadas al servicio no tienen relación alguna. RESTFul es el nombre utilizado para las aplicaciones que utilizan este tipo de arquitectura. REST posee otras bondades que lo diferencias de otras arquitecturas. Por ejemplo, a través de los métodos HTTP, facilita la comunicación entre un cliente HTTP con un servidor HTTP sin la necesidad de realizar complejas configuraciones, permitiendo establecer una estrecha relación entre las operaciones básicas de manejo de datos (CRUD: Create, Read, Update, Delete). Las aplicaciones REST están formadas por recursos, los cuales de identifican a través de una URI

(Uniform Resource Identifier). Los servicios REST son de fácil acceso gracias al uso de URIs sencillas e intuitivas con formato PATH, al igual que una URL de internet. (UNIR, Computación en el Servidor Web (ISW) - PER5 2017-2018 - Tema 6, 2018, págs. 16,17).

4.5. Gestor de base de datos: PostgreSQL

PostgreSQL es un poderoso sistema de base de datos relacional open source que utiliza y extiende el lenguaje SQL. Cuenta con más de 30 años de desarrollo, remontándose en sus orígenes como parte del proyecto POSTGRES en el año 1986 dentro de la Universidad de California en Berkeley. PostgreSQL se puede utilizar en todos los sistemas operativos y tiene potentes complementos, como la popular extensión de base de datos geoespaciales PostGIS. PostgreSQL es altamente extensible, ofreciendo la posibilidad al desarrollador de definir sus propios tipos de datos, desarrollar funciones personalizadas y escribir códigos en diferentes lenguajes de programación sin la necesidad de recompilar su base de datos. Intenta cumplir además con el standard SQL y muchas de las funciones requeridas son compatibles, cumpliendo con al menos 160 de las 179 características obligatorias para SQL a partir del lanzamiento de la versión 10 en octubre del 2017, donde a partir de ese momento ninguna de las bases de datos relacional cumple en su totalidad con el estándar (Development, 2018).

4.6. Entornos de desarrollo integrado

Según (UNIR, Plataformas de Desarrollo de Software (ISW) - PER5 2017-2018 - Tema 1, 2018, págs. 6,7)**,** los IDE (Integrated Development Environment), constituyen una aplicación de software que proporciona a los programadores una serie de funcionalidades comunes para el desarrollo del software. Las bondades ofrecidas por los IDE comprenden desde la escritura del programa hasta su ejecución, destacándose entre sus herramientas:

- ✓ Editor de código fuente para escribir el programa a través de un formato de texto acorde al lenguaje de programación seleccionado.
- ✓ Herramientas para la documentación para generar de manera (semi)-automática y cumpliendo los estándares la documentación del programa.
- ✓ Herramientas de depuración para ayudar a localizar e identificar los errores.
- ✓ Entorno de ejecución para ejecutar el código de la aplicación desarrollada.

4.6.1. Eclipse

Eclipse es un IDE gratuito para desarrolladores escrito principalmente en Java. Es uno de los IDE más utilizados para Java, permitiendo el desarrollo de aplicaciones para varias plataformas (aplicaciones de escritorio, aplicaciones móviles, aplicaciones web y software empotrado). El proyecto Eclipse está operable dentro de los sistemas operativos Windows, Mac OSX y Linux, bajo una licencia pública de Eclipse y de código abierto (open source). Eclipse está apoyado por una comunidad de desarrolladores que facilitan su documentación y desarrollan plugins útiles para el desarrollo de software en todas sus etapas. Se puede considerar como el principal IDE en general ya que permite la programación en otros lenguajes diferentes a Java, por ejemplo, C++, PHP, Python, JavaScript, entre otros. Entre sus plugins principales se encuentra Windows Builder para el desarrollo de interfaces gráficas, permitiendo además a los programadores adaptar el IDE a sus necesidades a través de su sistema extensible de plugins. Entre los IDE gratuitos o de pago basados en Eclipse podemos encontrar a MyEclipse y RAD de IBM (UNIR, Plataformas de Desarrollo de Software (ISW) - PER5 2017-2018 - Tema 2, 2018, pág. 5).

4.6.2. Visual Studio Code

Visual Studio Code es un ligero, pero potente editor de código fuente que se puede ejecutar dentro de los sistemas operativos Windows, MacOS y Linux. Incluye un soporte integrado para Node.js, JavaScript, TypeScript y variedad de extensiones para otros lenguajes como PHP, C#, C++, Java, Go, Python, entre otros (Microsoft, 2018).

Visual Studio Code incluye soporte de autocompletado a través de IntelliSense, una rica comprensión semántica, refactorización de código y excelentes herramientas para tecnologías web como HTML, CSS, y JSON. Se integra además con administradores de paquetes y repositorios, incluye tareas comunes que facilitan el trabajo cotidiano de los desarrolladores y se acopla con el control de versiones distribuido de GIT, ofreciendo herramientas para el análisis de las diferencias en el código fuente (Kahlert & Giza, 2016, pág. 4).

4.7. Herramienta CASE: Papyrus

Según (UNIR, Metodologías, Desarrollo y Calidad de la Ingeniería de Software (ISW) - PER5 2017-2018, 2018, pág. 20), la ingeniería de software asistida por ordenador, brindan soporte automatizado para gestionar las diferentes actividades de los procesos de software, por ejemplo, ingeniería de requisitos, diseño, desarrollo y pruebas.

Además, las herramientas CASE facilitan la construcción del sistema software a través de editores de diseño, diccionarios de datos, compiladores, depuradores, entre otros.

De acuerdo con (Sommerville, 2005, págs. 79,80), las clasificaciones de CASE facilitan a entender la variedad de herramientas CASE, ayudando así en las actividades del proceso del software, resumiéndose en las siguientes categorías:

- ✓ Herramientas: Ofrecer soporte a actividades individuales del proceso software, por ejemplo, la verificación de la consistencia del diseño y la compilación de un software.
- ✓ Workbenches: Contribuyen a las tareas del proceso como la especificación, el diseño, etc.
- ✓ Entornos: Ofrecen soporte a todos los procesos del software y generalmente incluyen varios bancos de trabajo.

Eclipse Papyrus es una plataforma de lenguaje específico de dominio (DSL) basada en el lenguaje de modelado estándar más extendido, el lenguaje de modelado unificado (UML). Esta aplicación de código abierto tiene dos objetivos principales. En primer lugar, pretende implementar la especificación UML completa, lo que le permite ser utilizada como la implementación de referencia para el estándar Object Management Group (OMG). En segundo lugar, tiene la intención de proporcionar una herramienta abierta, robusta, altamente escalable y altamente personalizable para definir las DSL y las herramientas correspondientes. Lo hace usando el mecanismo de perfil UML, así como las potentes funciones de personalización de la interfaz de usuario. Construido sobre Eclipse como un proyecto de código abierto, Papyrus es un candidato ideal para este propósito (Gérard, s.f.).

5. Desarrollo específico de la contribución

5.1. Requerimientos

(Sommerville, 2005, pág. 108) define el concepto de requerimientos como “La descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información”.

5.1.1. Requerimientos funcionales

De acuerdo con (Sommerville, 2005, pág. 109), los requerimientos funcionales “Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer”.

El sistema desarrollado cumple con los siguientes requerimientos funcionales:

1. El sistema permitirá a los usuarios autenticarse mediante usuario y contraseña.
2. El sistema permitirá a los usuarios cerrar sesión.
3. El sistema permitirá al usuario actualizar su contraseña.
4. El sistema permitirá al administrador visualizar los logs.
5. El sistema permitirá al administrador gestionar los usuarios.
6. El sistema enviará al usuario la contraseña inicial por correo electrónico.
7. El sistema permitirá al administrador gestionar los roles.
8. El sistema permitirá al administrador gestionar los perfiles.
9. El sistema permitirá al administrador gestionar los permisos.
10. El sistema permitirá al administrador resetear la contraseña de los usuarios.
11. El sistema permitirá al administrador bloquear y desbloquear a los usuarios.
12. El sistema permitirá al usuario gestionar las empresas.
13. El sistema permitirá al usuario gestionar los cultivos.
14. El sistema permitirá al usuario gestionar los productos.
15. El sistema permitirá al usuario agregar detalles a los productos.
16. El sistema permitirá al usuario administrar los contratos.
17. El sistema verificará en cada requerimiento funcional si el token de sesión enviado por el usuario es válido, excepto para el RF1, RF2 y RF3.

5.1.2. Requerimientos no funcionales

Según (Sommerville, 2005, págs. 109,110), los requerimientos no funcionales “Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad.”

El sistema desarrollado cumplirá con los siguientes requerimientos no funcionales:

1. Requerimiento de confiabilidad:
    a. Se debe garantizar que el sistema esté disponible las 24 horas del día.
    b. Las actividades de mantenimiento y supervisión del sistema no deben interferir en el correcto funcionamiento de la aplicación.
2. Requerimiento de seguridad:
    a. El sistema garantizará mantenerse en un estado seguro en caso de fallar algunas de sus funcionalidades.
    b. El sistema verificará si el usuario se encuentra autenticado antes de ejecutar alguna funcionalidad.
    c. El sistema verificará los permisos del usuario autenticado antes de ejecutar alguna funcionalidad.
    d. El sistema registrará las acciones solicitadas por los usuarios.
3. Requerimiento de portabilidad:
    a. El sistema debe permitir su despliegue tanto en la plataforma Windows como en Linux.
4. Requerimiento de apariencia o interfaz externa:
    a. El sistema debe validar todos los datos de entrada.
5. Restricciones de diseño e implementación:
    a. La implementación del sistema debe contar con un conjunto de patrones y principios de diseño que permita la reutilización de código, así como la facilidad de agregar nuevas funcionalidades.

5.1.3. Casos de uso

Según (Jacobson, Spence, & Bittner, CASOS DE USO 2.0: La guía para ser exitoso con los casos de uso, 2013, pág. 4), un caso de uso “expresa todas las formas de usar un sistema para alcanzar una meta particular para un usuario. En conjunto, los casos de uso le proporcionan todos los caminos útiles de usar el sistema e ilustran el valor que este provee.”

5.1.3.1. Diagrama de casos de uso

En la Ilustración 3 se muestran los actores que interactúan con el sistema y sus relaciones con los casos de uso principales. El actor “Administrador” ejecuta las acciones relacionadas a la seguridad mientras el actor “Usuario” las relacionadas al negocio propio en sí. A tal efecto no quiere decir que un usuario no pueda realizar acciones de seguridad sobre el sistema, pues mediante los casos de uso “Gestionar permisos” y “Gestionar roles” se le puede conceder a cualquier usuario permisos sobre cualquier acción.

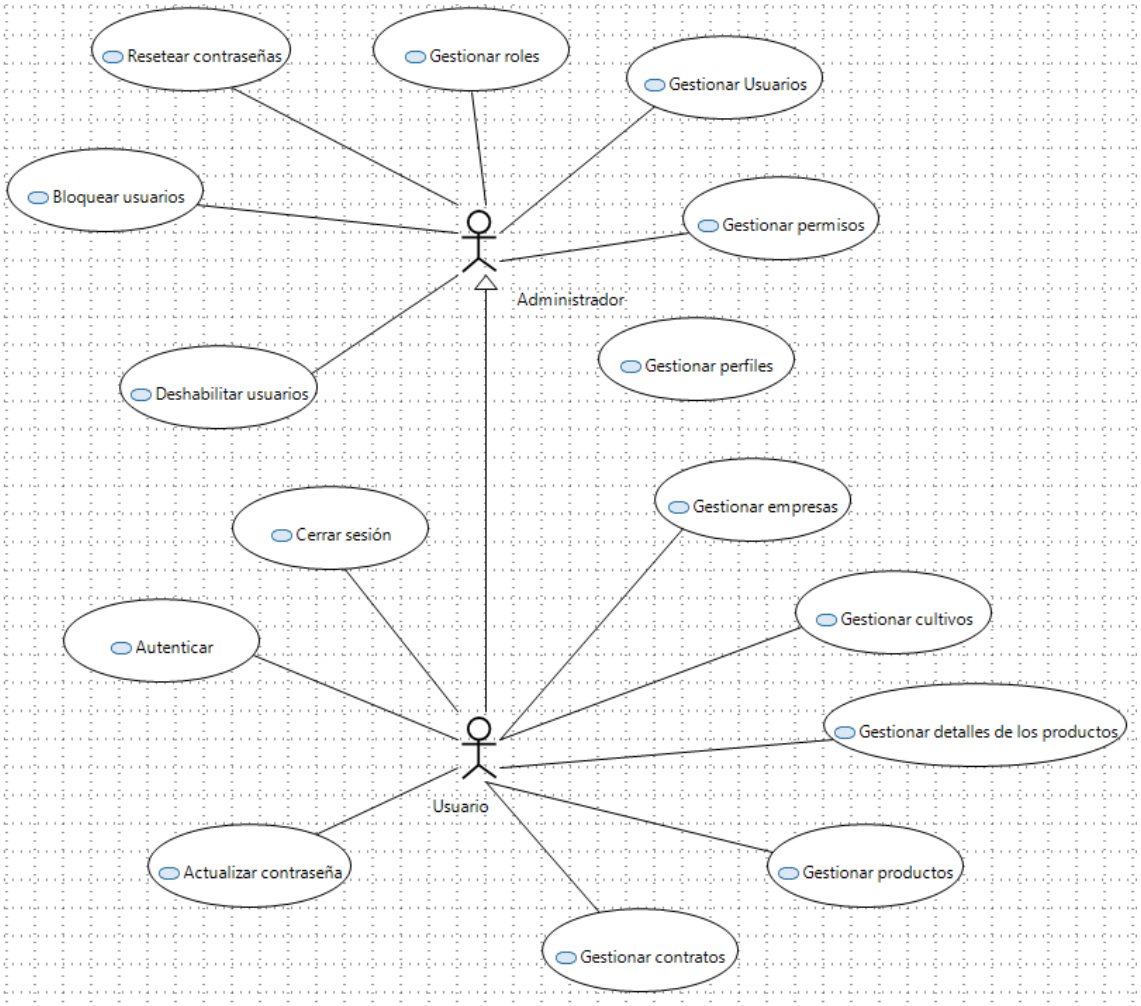


Ilustración 3: Diagrama de casos de uso

En la Ilustración 4 se detalla el caso de uso “Verificar sesión” y sus relaciones. Excepto los casos de uso “Autenticar”, “Cerrar sesión” y “Actualizar contraseña”, todos los casos de uso tienen una relación de <inlcude> con el caso de uso “Verificar sesión”, que a su vez tiene una relación de <include> con el caso de uso “Control de acceso”. El objetivo

del caso de uso “Verificar sesión” es validar que exista el token de seguridad enviado al cliente luego de su satisfactoria autenticación, donde el token debe ser enviado en la cabecera de cada petición realizada al servidor.

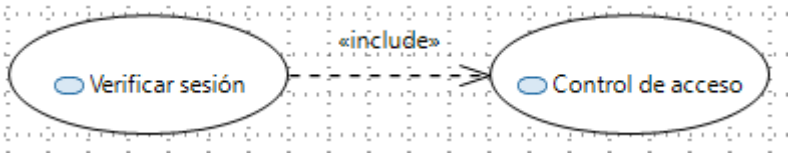


Ilustración 4: Caso de uso "Verificar sesión"

En la Ilustración 5 se muestra un ejemplo de las relaciones que participan en cada uno de los casos de uso de tipo <Gestionar>, por ejemplo, “Gestionar Usuario”, “Gestionar Productos” y “Gestionar Contratos”. En el diagrama general no se muestran todas las relaciones para su mejor compresión. El usuario tiene la decisión de ejecutar las acciones de adicionar, actualizar, eliminar o listar sobre cada caso de uso de tipo <Gestionar>, estableciendo una relación de <extend> sobre el caso de uso original. A su vez existe una relación de <include> sobre la acción de obtener un elemento antes de ser actualizado o eliminado.

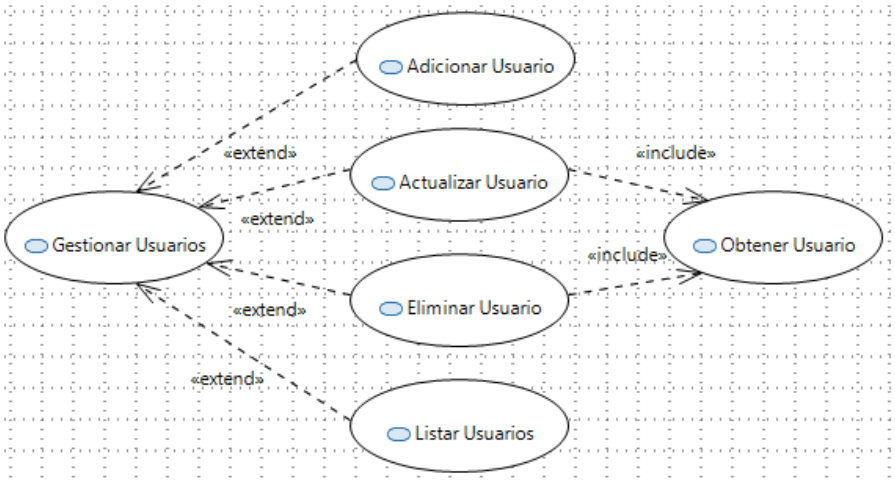


Ilustración 5: Casos de uso relacionados a los casos de uso tipo <Gestionar>

5.1.3.2. Descripción de los principales casos de uso

Tabla 2: Descripción del caso de uso "Autenticar usuario"

RF-01	**Autenticar usuario**	
Descripción	El sistema solicitará la autenticación al usuario mediante su contraseña para poder realizar cualquier acción sobre el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicitar autenticarse en el sistema.
	2	El sistema le muestra el formulario con los campos para ingresar el alias y la contraseña del usuario.
	3	El usuario ingresa su alias y su contraseña y selecciona la opción “Autenticar”.
	4	El sistema verifica que se hayan ingresado todos los datos.
	5	El sistema valida que los datos ingresados son correctos.
	6	El sistema verifica que el usuario no se encuentre bloqueado
	7	El sistema activa los accesos a los que puede ingresar el usuario dependiendo de los permisos configurados.
Excepciones	**Paso**	**Acción**
	4.1	Si el usuario no ha ingresado todos los datos solicitados, el sistema muestra un mensaje solicitándole al usuario que ingrese todos los datos obligatorios.
	5.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje detallando que el usuario o la contraseña del usuario no son correctas.
	6.1	Si el usuario se encuentra bloqueado el sistema mostrará un mensaje indicando los detalles.
	7.1	Si el usuario no tiene asociado ningún recurso, el sistema no habilitará ninguno de sus accesos.

Tabla 3: Descripción del caso de uso "Actualizar contraseña"

RF-03	**Actualizar contraseña**	
Descripción	El sistema solicitará al usuario la actualización de su contraseña al intentar autenticarse por primera vez.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario el formulario donde debe ingresar su contraseña actual, su nueva contraseña y la verificación de la nueva contraseña para evitar errores.
	2	El usuario ingresa todos los datos solicitados por el sistema y selecciona la opción “Actualizar contraseña”.

	3	El sistema verifica que todos los datos solicitados hayan sido ingresados.
	4	El sistema verifica que los datos ingresados se encuentren correctos.
	5	El sistema le muestra un mensaje al usuario informándole que su contraseña ha sido actualizada correctamente.
Excepciones	**Paso**	**Acción**
	3.1	Si el usuario no ha ingresado todos los datos solicitados, el sistema le muestra al usuario un mensaje solicitándole que ingrese todos los datos obligatorios.
	4.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje detallando que la fortaleza de la contraseña no es la indicada o que la nueva contraseña y su verificación no coinciden.

Tabla 4: Descripción del caso de uso "Adicionar rol"

RF-07	**Adicionar rol**	
Descripción	El usuario le solicitará al sistema agregar un nuevo rol.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario le solicita al sistema agregar un nuevo rol.
	2	El sistema muestra el formulario para ingresar los datos necesarios.
	3	El usuario ingresa los datos solicitados por el sistema y selecciona la opción “Agregar”.
	4	El sistema verifica que todos los datos solicitados se hayan ingresado.
	5	El sistema verifica que los datos ingresados son correctos.
	6	El sistema le muestra un mensaje al usuario informándole que el rol ha sido agregado correctamente.
	7	El sistema muestra la lista de roles actualizada con los datos del nuevo rol ingresado por el usuario.
Excepciones	**Paso**	**Acción**
	3.1	Si el usuario no ha ingresado todos los datos solicitados, el sistema le muestra al usuario un mensaje solicitándole que ingrese todos los datos obligatorios.
	4.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje detallando los diferentes motivos.

Tabla 5: Descripción del caso de uso "Adicionar usuario"

RF-05	**Adicionar usuario**	
Descripción	El usuario le solicitará al sistema agregar un nuevo usuario.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario le solicita al sistema agregar un nuevo usuario.
	2	El sistema muestra el formulario para ingresar los datos necesarios.
	3	El usuario ingresa los datos solicitados por el sistema y selecciona la opción “Agregar”.
	4	El sistema verifica que todos los datos solicitados se hayan ingresado.
	5	El sistema verifica que los datos ingresados son correctos.
	6	El sistema genera una contraseña aleatoria y es enviada al correo ingresado por el usuario.
	7	El sistema le muestra un mensaje al usuario informándole que el usuario ha sido agregado correctamente.
	8	El sistema muestra la lista de usuarios actualizada con los datos del nuevo usuario ingresado por el usuario.
Excepciones	**Paso**	**Acción**
	3.1	Si el usuario no ha ingresado todos los datos solicitados, el sistema le muestra al usuario un mensaje solicitándole que ingrese todos los datos obligatorios.
	4.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje detallando los diferentes motivos.
	6.1	Si el correo ingresado por el usuario no es correcto no podrá autenticarse la primera vez y debe solicitar al administrador que le actualice el correo y luego resetear su contraseña.

Tabla 6: Descripción del caso de uso "Gestionar permisos"

RF-09	**Gestionar permisos**	
Descripción	El usuario le solicitará al sistema administrar los recursos asociados a los roles.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona el rol al que desea actualizarle sus recursos.
	2	El usuario selecciona la opción “Administrar permisos”.

	3	El sistema muestra un listado de los recursos configurados en el sistema, apareciendo activados aquellos a los que el rol seleccionado tiene acceso en ese instante.
	4	El usuario actualiza los recursos asociados al rol seleccionado.
	5	El usuario selecciona la opción “Aceptar”.
	6	El sistema actualiza los recursos asociados al rol seleccionado.
	7	El sistema le muestra un mensaje al usuario informándole que el acceso a los recursos ha sido actualizado.
Excepciones	**Paso**	**Acción**
	3.1	Si el usuario no selecciona ningún recurso provocará que todos los usuarios relacionados al rol seleccionado no tendrán acceso alguno a las funcionalidades del sistema.

Tabla 7: Descripción del caso de uso "Gestionar perfiles"

RF-08	**Gestionar perfiles**	
Descripción	El usuario le solicitará al sistema administrar los usuarios asociados a los roles.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona el rol al que desea actualizarle sus usuarios asociados.
	2	El usuario selecciona la opción “Administrar perfiles”.
	3	El sistema muestra un listado de los usuarios registrados en el sistema, apareciendo activados aquellos a los que el rol seleccionado se encuentra asociado.
	4	El usuario actualiza los usuarios asociados al rol seleccionado.
	5	El usuario selecciona la opción “Aceptar”.
	6	El sistema actualiza los usuarios asociados al rol seleccionado.
	7	El sistema le muestra un mensaje al usuario informándole que los usuarios asociados al rol seleccionado se encuentran actualizados.
Excepciones	**Paso**	**Acción**
	3.1	Si no se selecciona ningún usuario provocará que todos los usuarios relacionados al rol seleccionado no tendrán acceso alguno a las funcionalidades del sistema.

Tabla 8: Descripción del caso de uso "Adicionar productos"

RF-14	**Adicionar productos**	
Descripción	El usuario le solicitará al sistema agregar un producto necesario posteriormente para los contratos.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario le solicita al sistema agregar un nuevo producto.
	2	El sistema le muestra el formulario con los campos necesarios para agregar el producto.
	3	El usuario ingresa los datos solicitados por el sistema. Entre los datos solicitados se encuentra el tipo de unidad de medida que debe encontrarse entre los siguientes valores: masa, capacidad y unidad.
	4	El usuario selecciona la opción “Agregar”.
	5	El sistema verifica que los datos ingresados por el usuario sean correctos.
	6	El sistema muestra un mensaje que producto ha sido agregado correctamente.
	7	El sistema muestra el listado de los productos, entre los que se encuentran el nuevo producto agregado recientemente.
Excepciones	**Paso**	**Acción**
	5.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje con el detalle de los errores.

Tabla 9: Descripción del caso de uso "Adicionar cultivos"

RF-13	**Adicionar cultivos**	
Descripción	El usuario le solicitará al sistema agregar un cultivo necesario posteriormente para los contratos.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario le solicita al sistema agregar un nuevo cultivo.
	2	El sistema le muestra el formulario con los campos necesarios para agregar el cultivo.
	3	El usuario ingresa los datos solicitados por el sistema.
	4	El usuario selecciona la opción “Agregar”.
	5	El sistema verifica que los datos ingresados por el usuario sean correctos.
	6	El sistema muestra un mensaje que cultivo ha sido agregado correctamente.

	7	El sistema muestra el listado de los cultivos, entre los que se encuentran el nuevo cultivo agregado recientemente.
Excepciones	**Paso**	**Acción**
	5.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje con el detalle de los errores.

Tabla 10: Descripción del caso de uso "Adicionar contratos"

RF-16	**Adicionar contratos**	
Descripción	El usuario le solicitará al sistema agregar un nuevo contrato.	
Precondición	Usuario autenticado correctamente en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario le solicita al sistema agregar un nuevo contrato.
	2	El sistema le muestra el formulario con los campos necesarios para agregar el contrato. Entre los campos que debe seleccionar el usuario se encuentran el producto y el cultivo. Se pueden seleccionar varios cultivos y por cada cultivo seleccionado se pueden seleccionar varios productos.
	3	El usuario ingresa los datos solicitados
	4	El usuario selecciona la opción “Agregar”.
	5	El sistema verifica que los datos ingresados por el usuario sean correctos.
	6	El sistema muestra un mensaje que el contrato ha sido agregado correctamente.
	7	El sistema muestra el listado de los contratos, entre los que se encuentran el nuevo contrato agregado recientemente.
Excepciones	**Paso**	**Acción**
	5.1	Si los datos ingresados por el usuario no son correctos, el sistema mostrará un mensaje con el detalle de los errores.

5.2. Análisis y diseño

5.2.1. Modelo relacional de la base de datos

En la Ilustración 6, se muestra el modelo relacional de la base de datos utilizada en el aplicativo, detallándose las tablas y campos, sus relaciones, así como las llaves primarias y foráneas.

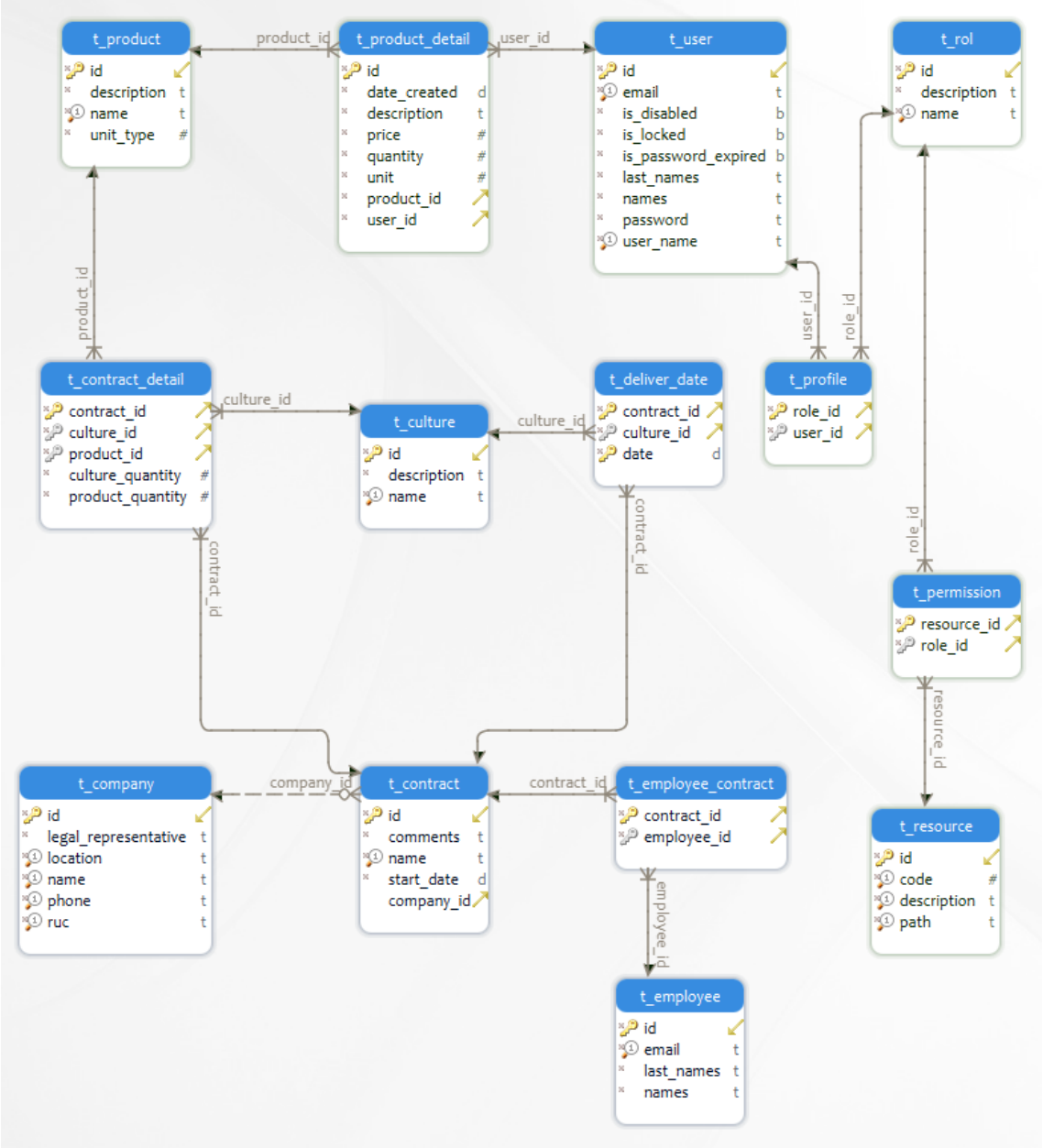


Ilustración 6: Modelo relacional

Tabla 11: Descripción de la tabla t_user

Nombre: t_user **Descripción:** Tabla que almacena la información de los usuarios registrados en el sistema.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
user_name	varchar	32	Alias (único) del usuario.
password	varchar	128	Contraseña encriptada del usuario.

names	varchar	64	Nombres del usuario.
last_names	varchar	64	Apellidos del usuario.
email	varchar	64	Correo electrónico (único) del usuario.
is_disabled	boolean		Define si el usuario está habilitado o no.
is_locked	boolean		Define si el usuario está bloqueado o no.
is_password_expired	boolean		Define si la contraseña se encuentra expirada o no.

Tabla 12: Descripción de la tabla t_rol

Nombre: t_rol **Descripción:** Tabla que almacena la información de los roles.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
name	varchar	32	Nombre (único) del rol.
description	varchar	128	Descripción del rol.

Tabla 13: Descripción de la tabla t_resource

Nombre: t_resource **Descripción:** Tabla que almacena la información de los recursos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
code	integer		Código (único) del recurso.
description	varchar	512	Descripción del recurso.
path	varchar	256	Ruta (única) de acceso del recurso.

Tabla 14: Descripción de la tabla t_profile

Nombre: t_profile **Descripción:** Tabla que almacena la información de los perfiles (usuarios asociados a los roles).			
Campo	**Tipo**	**Tamaño**	**Descripción**
role_id	uuid		Llave foránea relacionada a la tabla t_rol.
user_id	uuid		Llave foránea relacionada a la tabla t_user.

Tabla 15: Descripción de la tabla t_permission

Nombre: t_permission

Descripción: Tabla que almacena la información de los permisos (roles asociados a los recursos).			
Campo	**Tipo**	**Tamaño**	**Descripción**
resource_id	uuid		Llave foránea relacionada a la tabla t_resource.
role_id	uuid		Llave foránea relacionada a la tabla t_rol.

Tabla 16: Descripción de la tabla t_product

Nombre: t_product **Descripción:** Tabla que almacena la información de los productos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
name	varchar	32	Nombre del producto.
description	varchar	128	Descripción del permiso.
unit_type	integer		Tipo de unidad de medida del producto.

Tabla 17: Descripción de la tabla t_culture

Nombre: t_culture **Descripción:** Tabla que almacena la información de los cultivos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
name	varchar	32	Nombre del cultivo.
description	varchar	128	Descripción del cultivo.

Tabla 18: Descripción de la tabla t_company

Nombre: t_company **Descripción:** Tabla que almacena la información de las empresas.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
legal_representative	varchar	32	Representante legal de la empresa.
name	varchar	32	Nombre de la empresa.
location	varchar	64	Ubicación de la empresa.
phone	varchar	32	Teléfono de la empresa.
ruc	varchar	13	RUC de la empresa.

Tabla 19: Descripción de la tabla t_contract

Nombre: t_contract **Descripción:** Tabla que almacena la información de los contratos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
start_date	timestamp		Fecha de inicio del contrato
company_id	uuid		Llave foránea relacionada a la tabla t_company.

Tabla 20: Descripción de la tabla t_product_detail

Nombre: t_product_detail **Descripción:** Tabla que almacena la información de los contratos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
id	uuid		Llave primaria.
date_created	timestamp		Fecha en que se adicionó el detalle.
description	varchar	512	Descripción del detalle agregado.
price	numeric	(10, 2)	Precio de compra.
quantity	numeric		Cantidad adquirido.
unit	integer		Unidad de medida.
product_id	uuid		Llave foránea relacionada a la tabla t_product.
user_id	uuid		Llave foránea relacionada a la tabla t_user.

Tabla 21: Descripción de la tabla t_contract_detail

Nombre: t_contract_detail **Descripción:** Tabla que almacena la información de los contratos.			
Campo	**Tipo**	**Tamaño**	**Descripción**
contract_id	uuid		Llave foránea relacionada a la tabla t_contract.
culture_id	uuid		Llave foránea relacionada a la tabla t_culture.
product_id	uuid		Llave foránea relacionada a la tabla t_product.
culture_quantity	integer		Cantidad de cultivos empleados.
product_quantity	integer		Cantidad de productos empleados.

5.2.2. Diagrama de clases

5.2.2.1. Diagrama de clases (frontend)

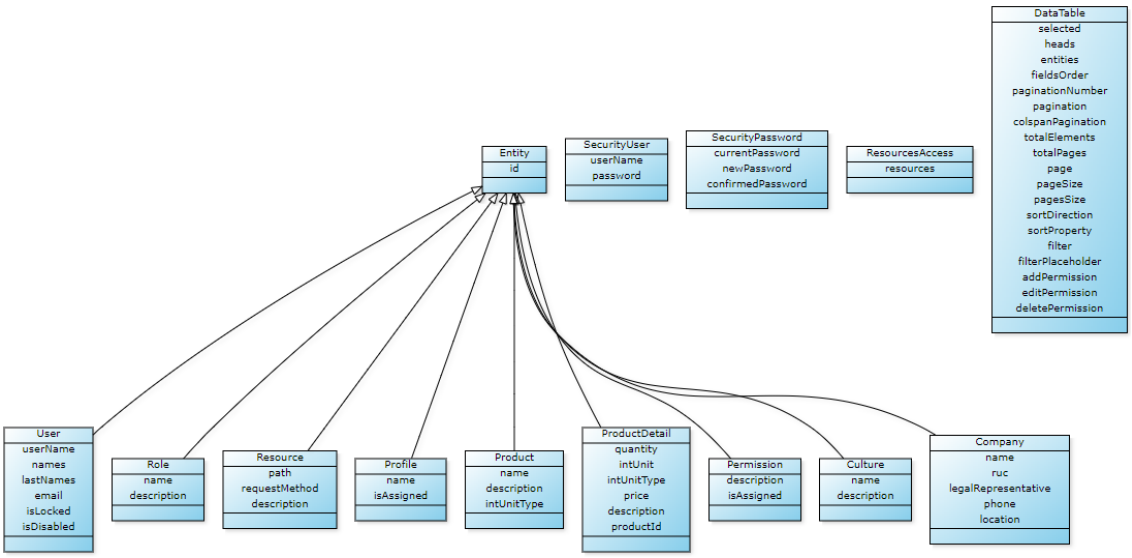


Ilustración 7: Diagrama de clases base

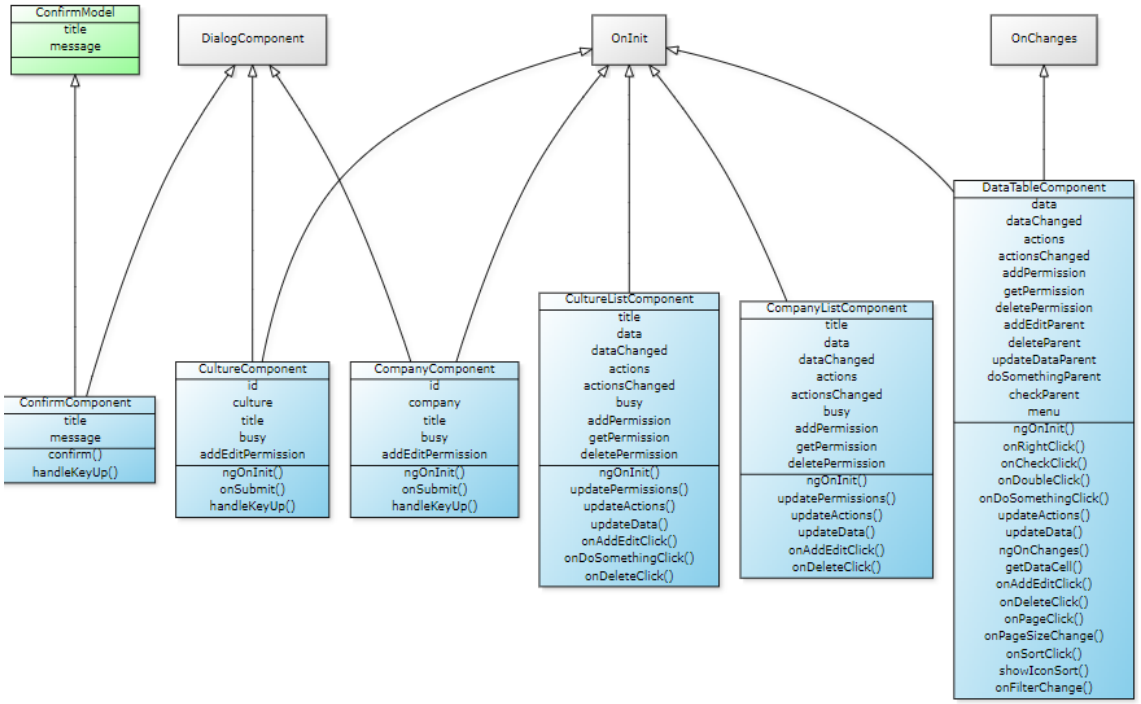


Ilustración 8: Diagrama de clases de los componentes (1ra parte)

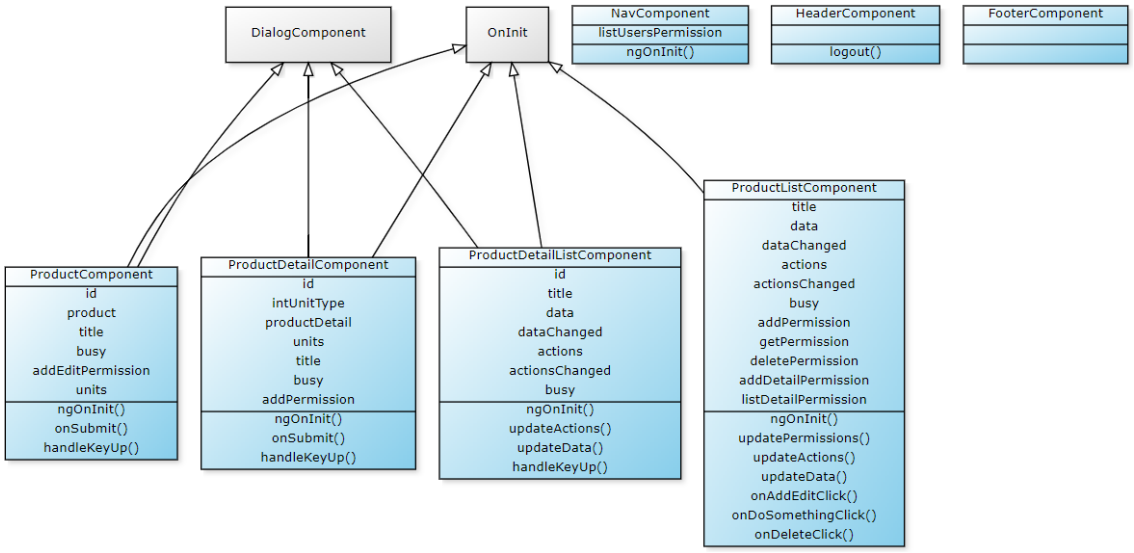


Ilustración 9: Diagrama de clases de los componentes (2da parte)

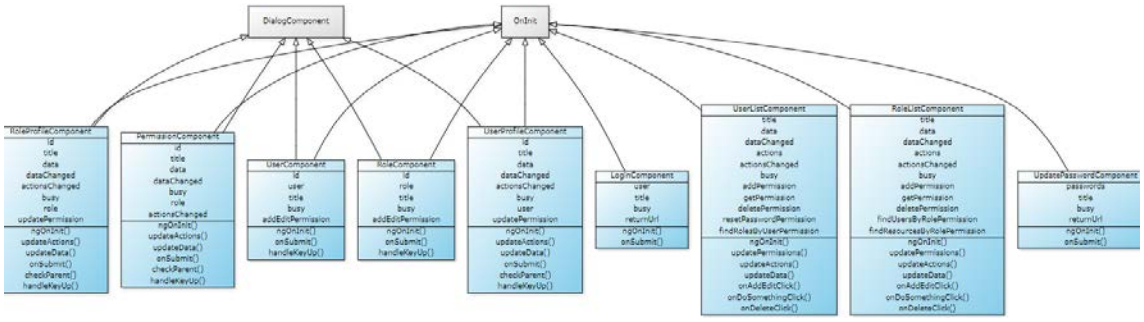


Ilustración 10: Diagrama de clases de los componentes (3ra parte)

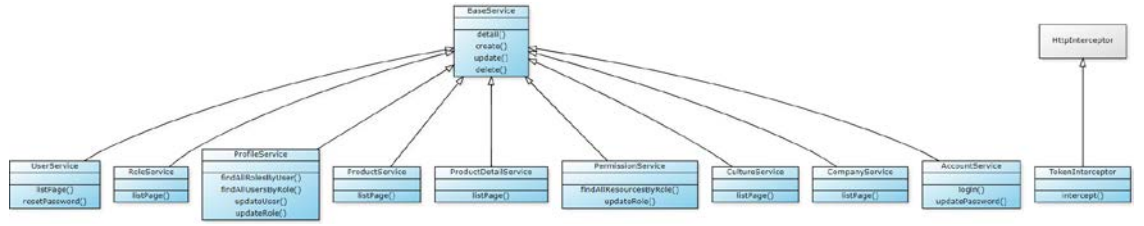


Ilustración 11: Diagrama de clases de los servicios

5.2.2.2. Diagrama de clases (backend)

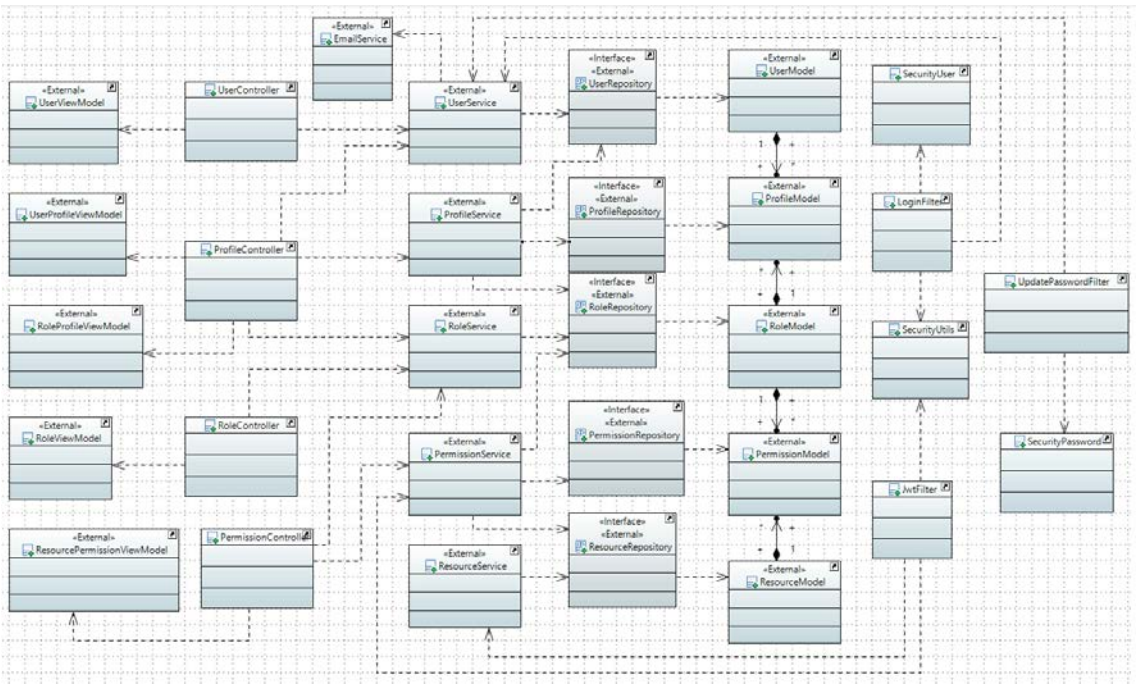


Ilustración 12: Diagrama de clases que intervienen en la gestión de la seguridad

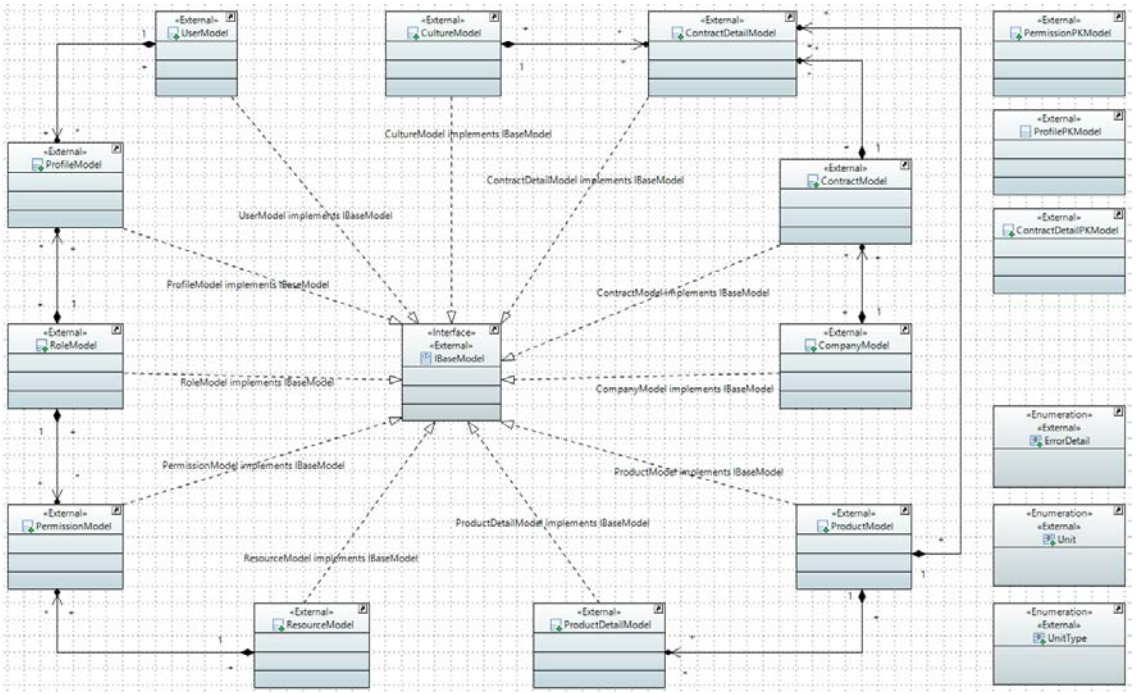


Ilustración 13: Diagrama de clases base

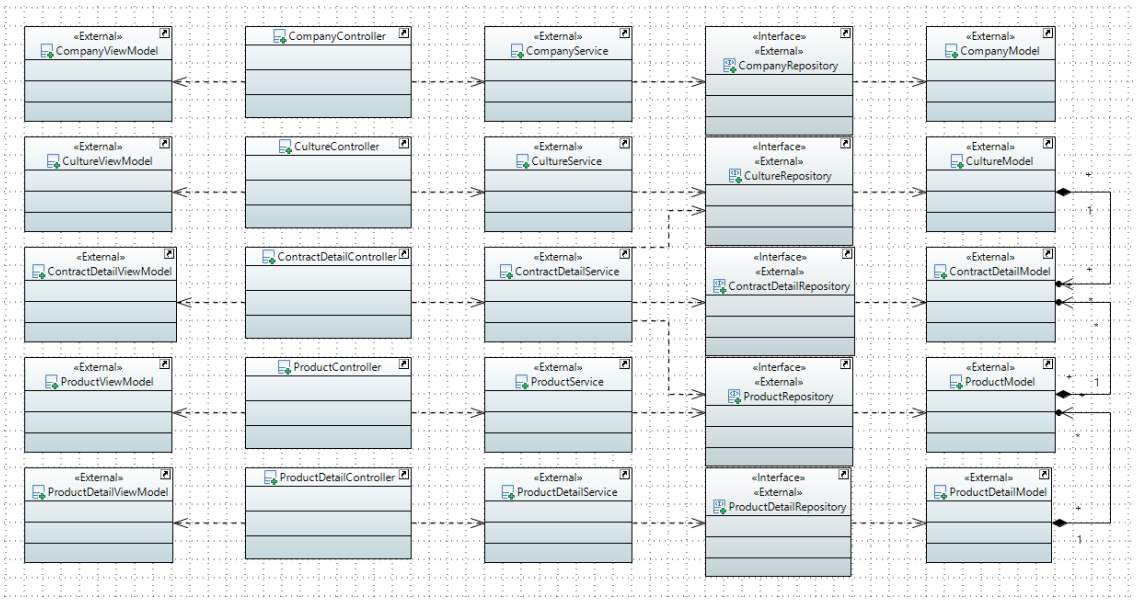


Ilustración 14: Diagrama de clases que intervienen en la gestión de los contratos

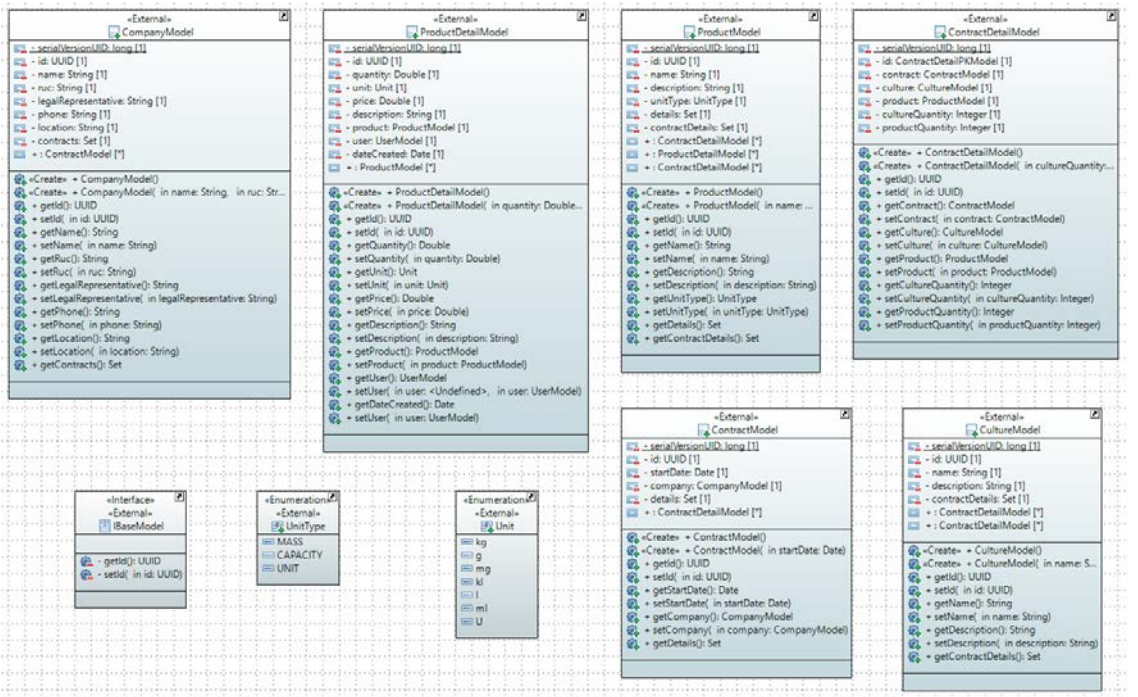


Ilustración 15: Detalles de las clases base

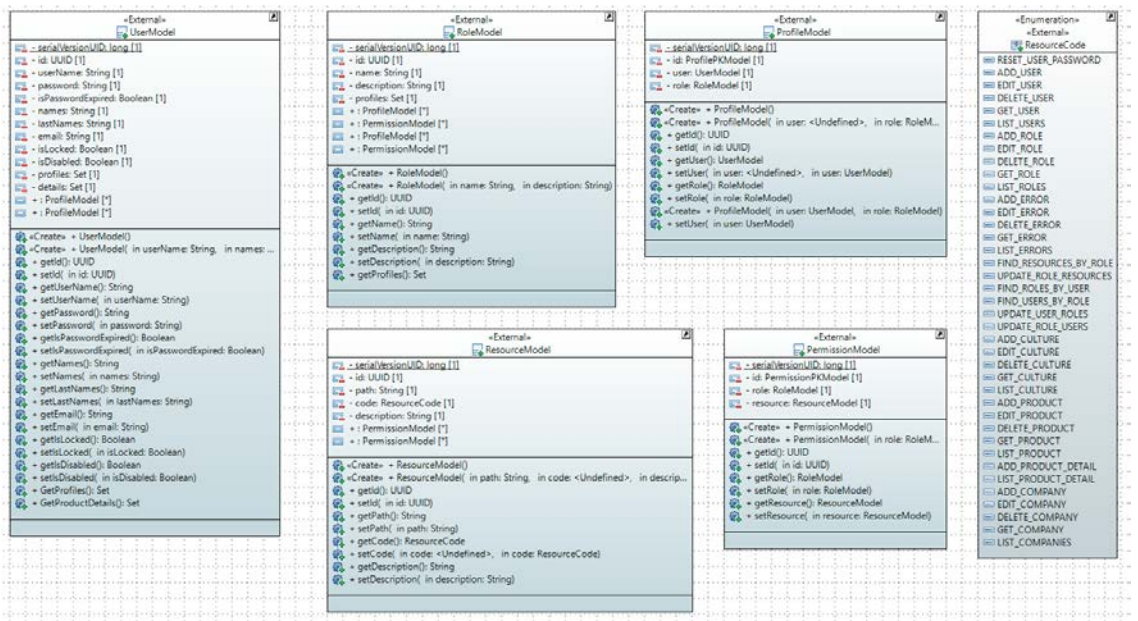


Ilustración 16: Detalles de las clases que intervienen en la gestión de la seguridad

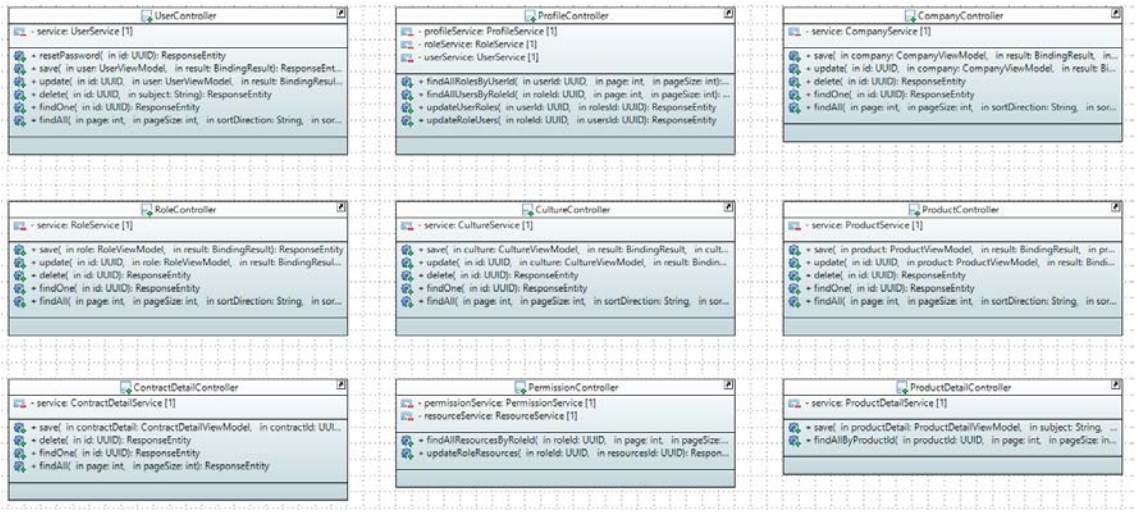


Ilustración 17: Detalles de las clases de tipo <Controller>

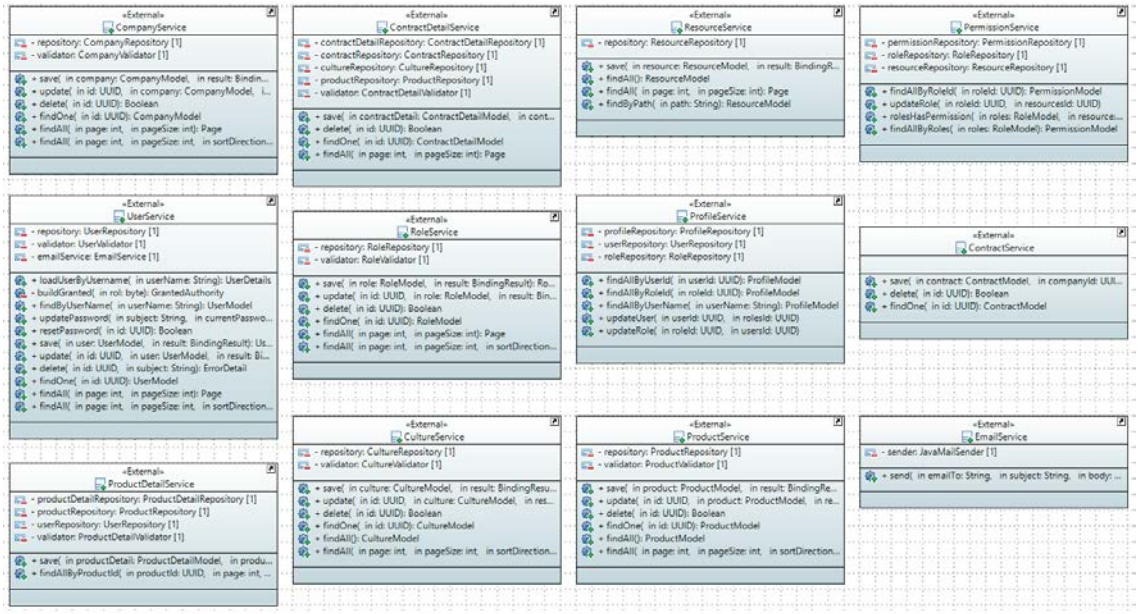


Ilustración 18: Detalles de las clases de tipo <Service>

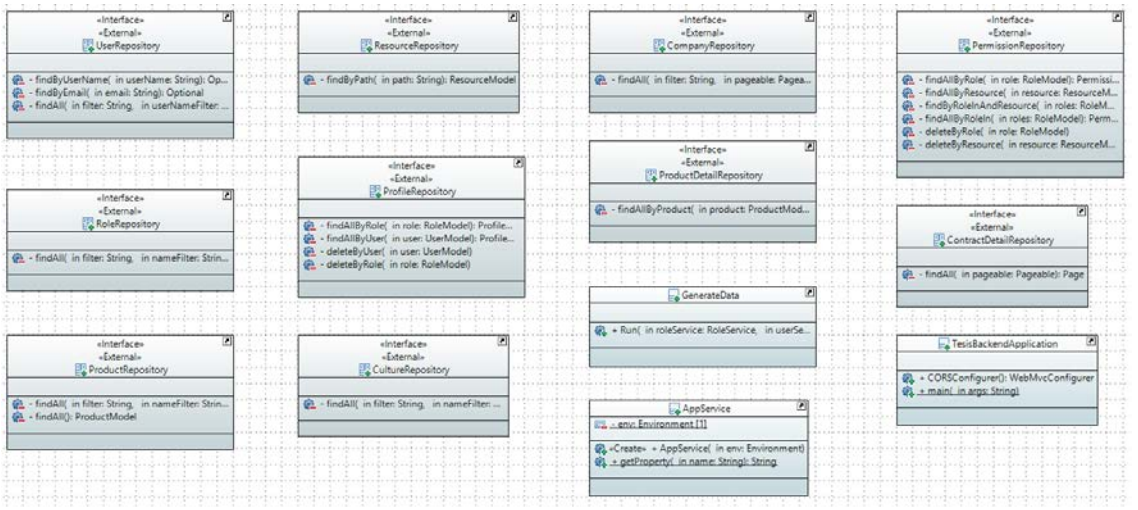


Ilustración 19: Detalles de las clases de tipo <Repository>

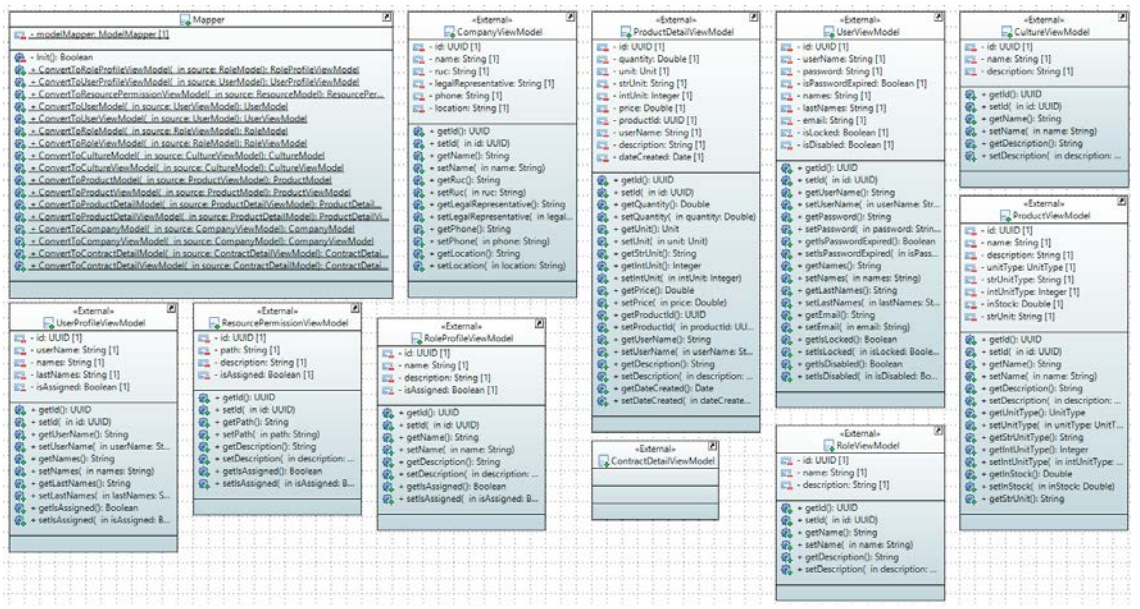


Ilustración 20: Detalles de las clases de tipo <ViewModel>

5.2.3. Diagrama de paquetes

En la Ilustración 21 se muestran los diferentes paquetes de software que contiene el backend del aplicativo y sus diferentes relaciones.

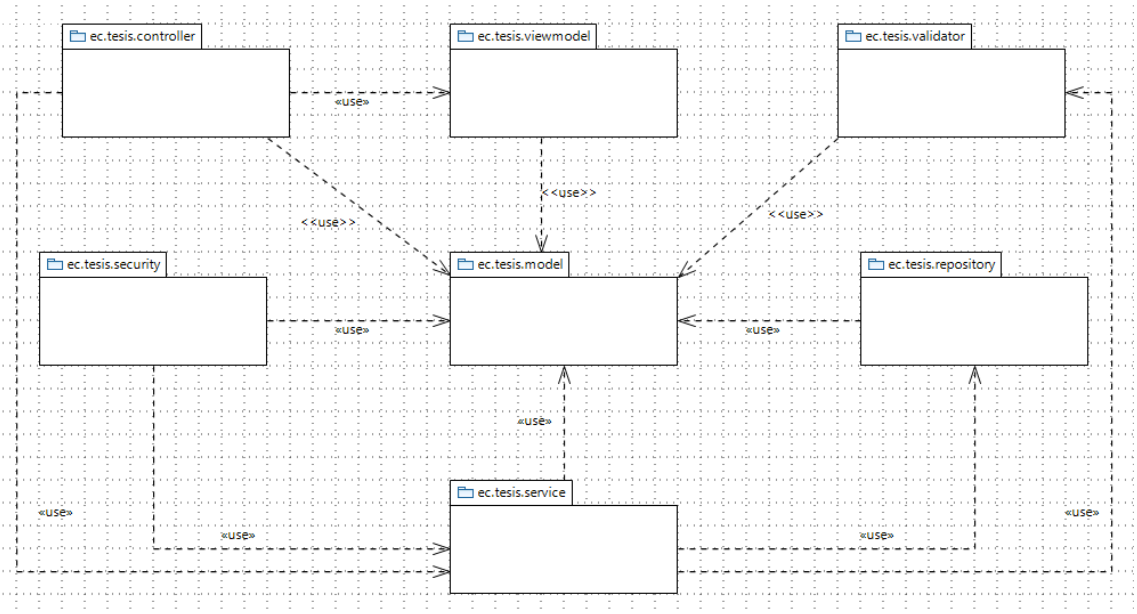


Ilustración 21: Diagrama de paquetes

Tabla 22: Descripción de los paquetes del backend

Nombre	Dependencias	Descripción
ec.tesis.model		Paquete donde se definen las clases del modelo que se utilizará en el aplicativo. Entre las características que se detallan en cada una de las clases se encuentran: nombre de la tabla, columnas, características de los columnas y relaciones entre tablas de la base de datos. Todas sus clases heredan de la clase <Serializable>.
ec.tesis.repository	ec.tesis.model	Paquete compuesto por las interfaces de acceso a datos. Se concretan las diferentes llamadas al base de datos, aunque en su mayoría son pocas debido su facilidad de acceso mediante la extensión con la clase <JpaRepository>. Entre sus métodos más utilizados podemos mencionar: findAll, findAllById, saveAll, getOne, entre otros.
ec.tesis.validator	ec.tesis.model	Paquete que concreta las clases que validan la entrada de datos a través de los servicios antes de su llegada a los repositorios. Llenan una estructura de datos que puede contener varios errores y sus detalles al mismo tiempo.
ec.tesis.service	ec.tesis.model, ec.tesis.repository, ec.tesis.validator	Paquete que definen las clases intermedias entre los controladores y los repositorios, facilitando su reutilización.

ec.tesis.viewmodel	ec.tesis.model	Paquete que contienen las clases con las estructuras necesarias para ser enviadas y obtenidas desde el frontend. Las clases del modelo en su mayoría no coinciden con la forma en que los datos son manipulados por el frontend. Existe una clase encargada de mapear las clases del modelo a las clases del viewmodel y viceversa.
ec.tesis.controller	ec.tesis.model, ec.tesis.service, ec.tesis.viewmodel	Paquete que denominan las clases de acceso al backend desde el frontend. Establecen una ruta única de acceso para cada uno de sus métodos. Todos sus métodos tienen tratamiento de errores a través del try-catch.
ec.tesis.security	ec.tesis.model, ec.tesis.service	Paquete donde se configuran los diferentes mecanismos de seguridad a través de la dependencia del paquete <spring-security>. Entre las funcionalidades configuradas se encuentran: la ruta de autenticación, la ruta de la actualización de la contraseña, los métodos utilizados durante la autenticación, mecanismo de generación de token tras una autenticación satisfactoria y método de verificación de token necesario en cada uno de los accesos al aplicativo.

5.2.4. Patrones de diseño

Debido a los problemas comunes dentro del desarrollo de aplicaciones, han aparecido diferentes soluciones reutilizables y probadas que han sido empleadas como plantilla por multitud de programadores. A esos pequeños códigos comunes se les conoce como patrones de diseño.

En dependencia de la naturaleza de la aplicación que se desee desarrollar se pueden aplicar varios patrones, pero siempre analizando el problema que se desea resolver y las ventajas de utilizar los patrones como solución a dichos problemas. No es obligatorio conocerlos todos, pero mientras más se dominen mayor robustez se obtendrá en el código implementado. Existen variedades de patrones de diseños, y continúan apareciendo nuevos debido al desarrollo vertiginoso de las tecnologías de desarrollo. Los patrones de diseño se pueden clasificar en creacionales, de comportamiento y estructurales.

En el desarrollo de la aplicación se utilizaron los siguientes patrones de diseño:

- ✓ Inyección de dependencia: Utilizado para ofrecerle a las clases objetos ya creados en lugar de construirlos por la propia clase. Entre sus principales ventajas podemos encontrar que facilita el diseño y ejecución de pruebas, tratando a la clase y al objeto inyectado como componentes diferentes.
    - o En Java se implementa mediante la palabra reservada <@Autowired>.

```
@Autowired
private CompanyRepository repository;

@Autowired
private CompanyValidator validator;
```

Ilustración 22: Implementación del patrón de diseño "Inyección de dependencia" en Java

    - o En angular mediante la palabra reservada <providers>.

```
@Component({
    selector: 'tesis-frontend-permission',
    templateUrl: '../../views/permission/permission.html',
    providers: [RoleService, PermissionService]
})
```

Ilustración 23: Implementación del patrón de diseño "Inyección de dependencia" en Angular

- ✓ Repository: Facilita el acceso a la base de datos a través de interfaces definidas.
  - o En Java entre los mecanismos utilizados podemos mencionar la herencia con la interfaz <JpaRepository>

```
@Repository
public interface RoleRepository extends JpaRepository<RoleModel, UUID>
```

Ilustración 24: Implementación del patrón de diseño "Repository" en Java

- ✓ Command: Permite encapsular las peticiones como objetos.

```
public class EmailService {

    @Autowired
    private JavaMailSender sender;


    public void exec(String emailTo, String subject, String body) {
        if (!Boolean.parseBoolean(AppService.getProperty("spring.mail.activated")))
            return;

        MimeMessagePreparator message = newMessage -> {
            newMessage.setRecipient( Message.RecipientType.TO, new InternetAddress(emailTo));
            newMessage.setFrom("from.email@gmail.com");
            newMessage.setSubject(subject);
            newMessage.setText(body);
        };

        sender.send(message);
    }

}
```

Ilustración 25: Implementación del patrón de diseño "Command" en Java

5.3. Implementación y pruebas

5.3.1. Diagrama de despliegue

En Ilustración 26, se muestran los diferentes nodos que intervienen en la solución entregada.

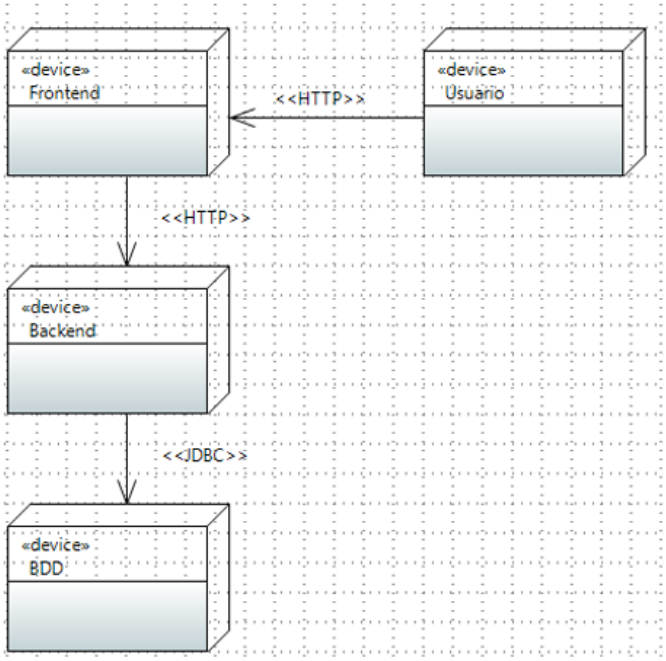


Ilustración 26: Diagrama de despliegue

Tabla 23: Descripción de los nodos del diagrama de despliegue

Nombre	Comunicación	Descripción
Base de datos (BDD)		Instancia del servidor de base de datos, utilizando como gestor a Postgres.
Backend (servidor de aplicaciones)	Base de datos	Instancia del servidor de aplicaciones, comunicándose directamente al servidor de base de datos. Se utiliza a Java como lenguaje de programación. Ofrece servicio a todo cliente que se comunique independientemente de su tecnología (web, móvil, otros backend). Valida siempre los mecanismos de seguridad.
Frontend (servidor web)	Backend	Instancia del servidor web. Es la interfaz que utiliza el usuario para acceder a las diferentes funcionalidades del aplicativo. Está compuesto por ventanas gráficas de fácil utilización y diseño

		agradable. Se utiliza a JavaScript como lenguaje base en su programación.
Usuario (máquina cliente)	Frontend	Instancias de las máquinas clientes utilizadas por los usuarios para interactuar con el aplicativo mediante los navegadores web.

5.3.2. Pruebas

En las tablas Tabla 24, Tabla 25 y Tabla 26 se muestran algunas de las pruebas realizadas al sistemas con el objetivo de verificar su correcto funcionamiento. Se realizaron las pruebas con una herramienta externa diferente al frontend desarrollado, para comprobar que existan validaciones por parte del backend. En caso de solo encontrarse validaciones por parte del frontend, el sistema se vería comprometido, y podría ser atacado por otros clientes.

Tabla 24: Pruebas al caso de uso "Autenticar"

Información General	
Identificador de caso de uso:	CU-01
Nombre de caso de uso:	Autenticación de usuario mediante su contraseña
Descripción Prueba:	Verificar que los usuarios se puedan autenticar con su contraseña.
Responsable:	Equipo de pruebas
Prerrequisitos	
El usuario no debe estar autenticado.	
Descripción de Casos de Prueba	
1- Verificar que los usuarios se puedan autenticar con su contraseña. 2- Verificar que los usuarios no registrados no se puedan autenticar. 3- Verificar que los usuarios con contraseña incorrecta no se puedan autenticar. 4- Verificar que los usuarios con la contraseña expirada el sistema le solicite actualizar la contraseña. 5- Verificar que los usuarios bloqueados o deshabilitados no se puedan autenticar.	

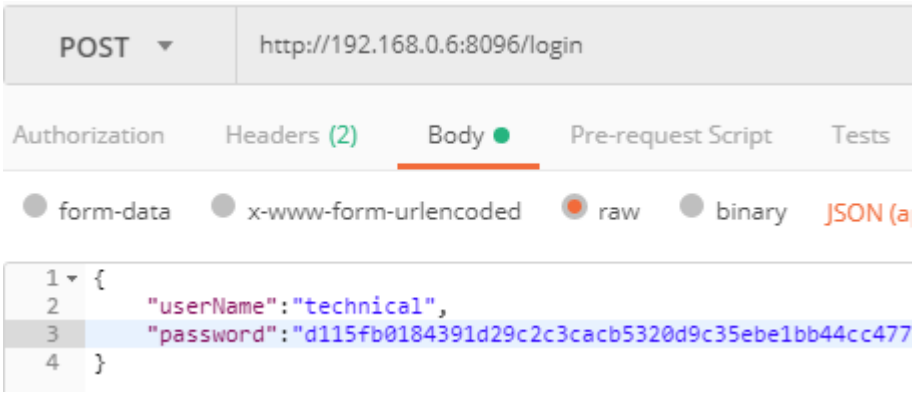
6- Verificar que al cliente donde se autentiquen los usuarios el servidor envíe un token para poder acceder a las funcionalidades del sistema.
7- Verificar que al cliente donde se autentiquen los usuarios el servidor envíe un listado de los recursos a los que puede acceder el usuario.

Servicio (s) Probados:

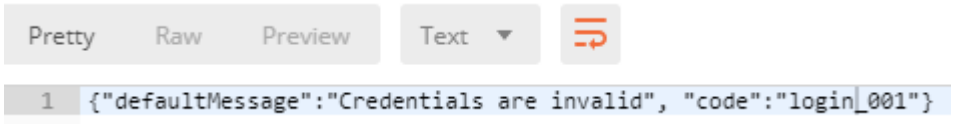
```
public Authentication attemptAuthentication(HttpServletRequest req, HttpServletResponse res)
```

Instrucciones de Prueba

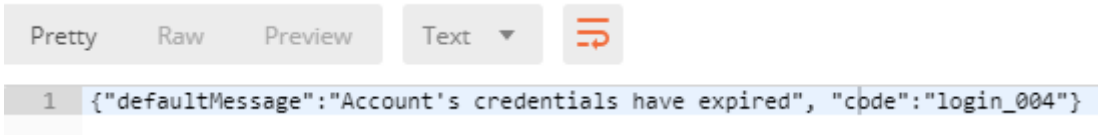
1. Acceder al backend a la siguiente url: http://ip-servidor/login mediante algún programa que permita el envío de peticiones HTTP REST, por ejemplo, Postman
2. Se debe configurar además el método POST de acceso y enviar los datos mediante las variables “userName” y “password”. La contraseña debe enviarse encriptada mediante el algoritmo “sha512”.



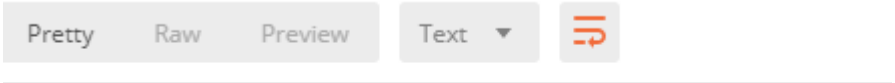
2.1. Si se envían los datos de usuario y contraseña correctos el sistema debe retornar el estado 200 (OK).
2.2. Si se envían los datos de un usuario no registrado o la contraseña incorrecta el sistema debe devolver el código de error “login_001”, indicando que las credenciales no son válidas.

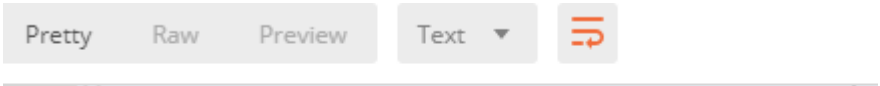


2.3. Si el usuario que se desea autenticar se encuentra con la contraseña expirada, el sistema debe devolver el código “login_004”, con el detalle del error.

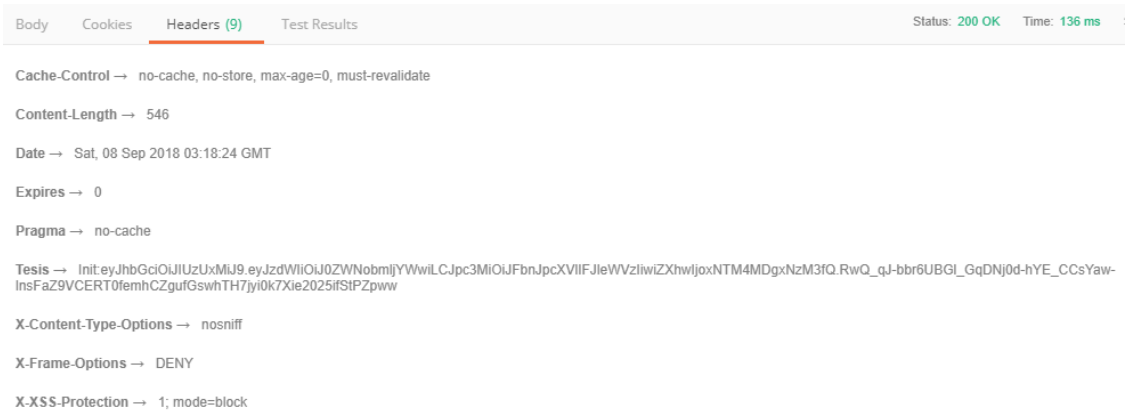


2.4. Si el usuario que intenta autenticarse se encuentra bloqueado o deshabilitado, el sistema lo debe informar mediante los códigos de error “login_002” y “login_003”.





2.5. Tras una autenticación satisfactoria el sistema debe enviar un token con la llave “Tesis”.



2.6. Luego de una correcta autenticación el sistema debe enviar el listado de recursos a los que puede acceder el usuario. El listado enviado solo se utiliza para que el frontend muestre el estado de sus controles actualizados, pero no con fines de seguridad. Si el usuario desea acceder a un recurso que no le es enviado el sistema verifica si tiene acceso o no.

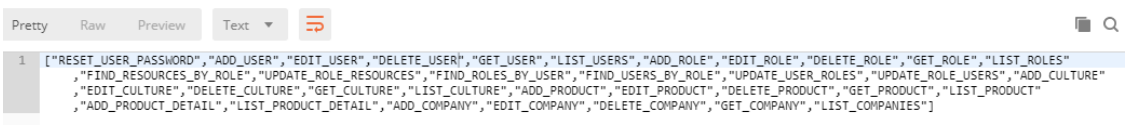


Tabla 25: Pruebas al caso de uso "Verificar sesión"

Información General

Identificador de caso de uso:	CU-17
Nombre de caso de uso:	Verificar sesión
Descripción Prueba:	El sistema debe verificar en cada solicitud al sistema que la sesión del usuario sea válida y que tenga acceso al recurso solicitado.
Responsable:	Equipo de pruebas

Prerrequisitos

Descripción de Casos de Prueba

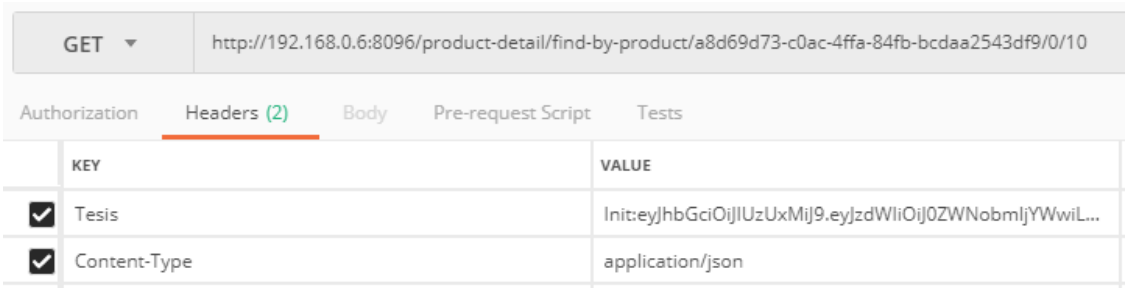
1- Verificar que los usuarios autenticados puedan acceder al recurso solicitado.
2- Verificar que los usuarios con la contraseña expirada no puedan acceder al recurso solicitado.
3- Verificar que los usuarios bloqueados o deshabilitados no puedan acceder al recurso solicitado.
4- Verificar que el token enviado sea válido.
5- Verificar que el recurso solicitado exista.
6- Verificar que el usuario tiene acceso al recurso solicitado.

Servicio (s) Probados:

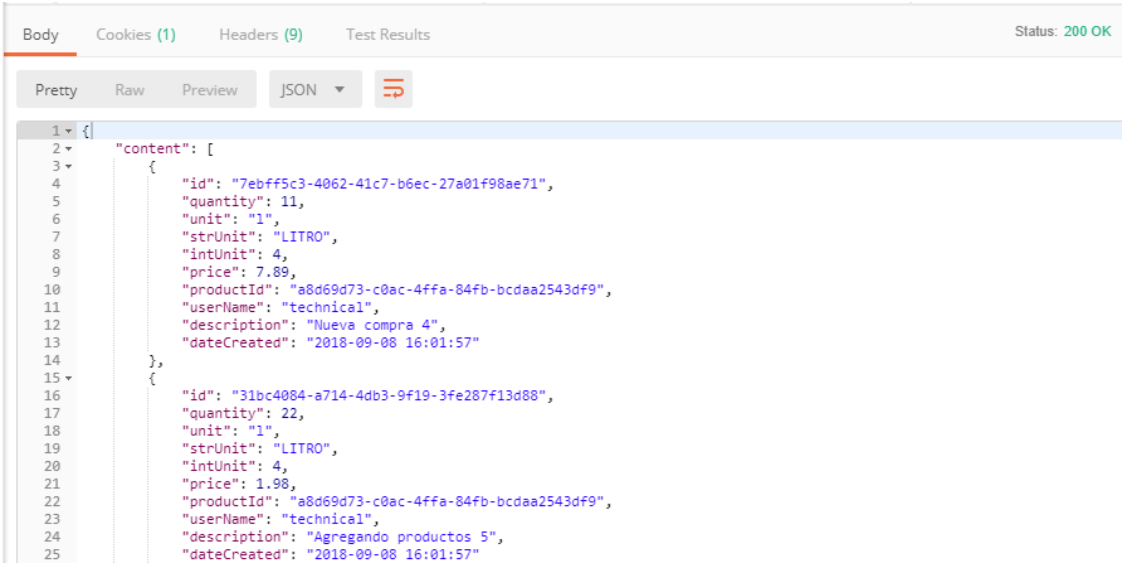
```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
```

Instrucciones de Prueba

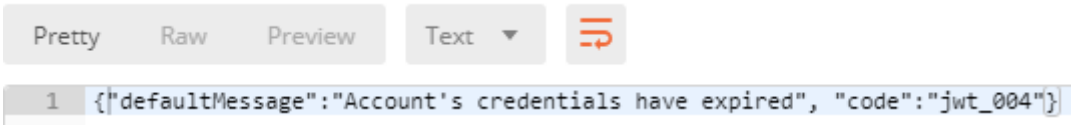
1. Acceder al backend a la siguiente url: http://ip-servidor:8096/product-detail/find-by-product/a8d69d73-c0ac-4ffa-84fb-bcdaa2543df9/0/10 mediante algún programa que permita el envío de peticiones HTTP REST, por ejemplo, Postman.
2. Se debe configurar además el método GET de acceso y agregar en la cabecera los datos del token enviado luego de la autenticación.



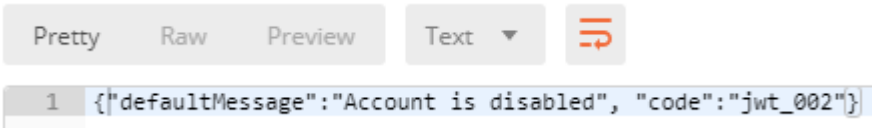
2.1. Si los datos del token son correctamente agregados y se accede a la dirección correcta el sistema debe retornar los datos solicitados mediante JSON y con estado 200 (OK).

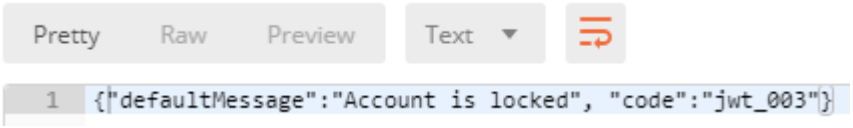


2.2. Si el usuario que se desea autenticar se encuentra con la contraseña expirada, el sistema lo debe informar a través del código “jwt_004”.

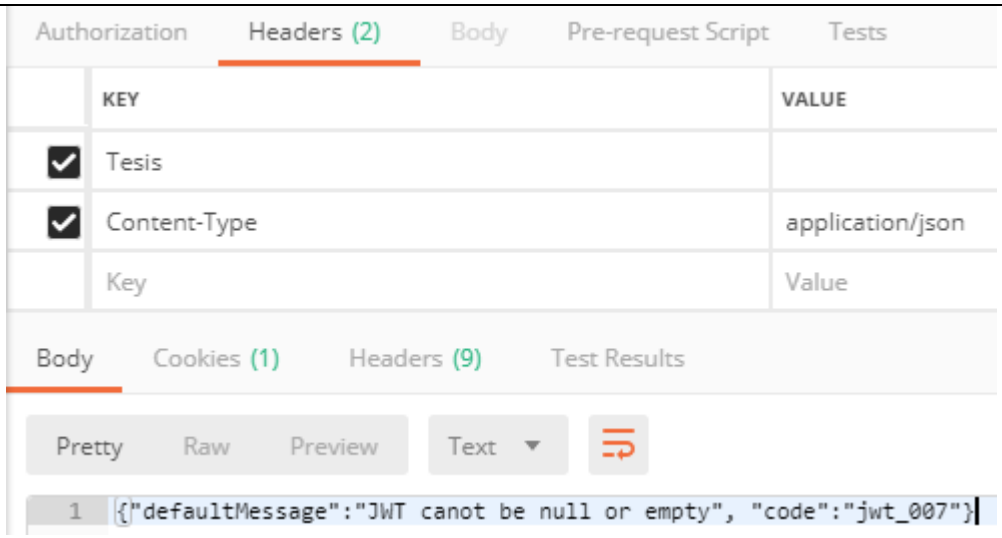


2.3. Si el usuario que intenta autenticarse se encuentra bloqueado o deshabilitado, el sistema lo debe informar a través de los códigos “jwt_002” y “jwt_003”.

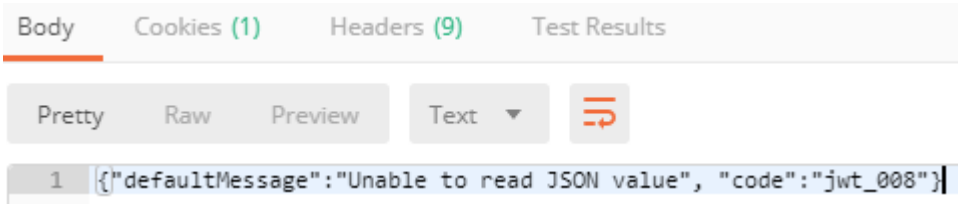




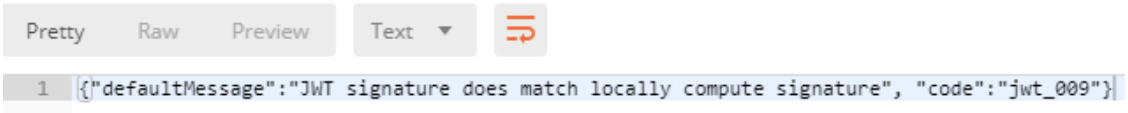
2.4. Si el token no es enviado, el sistema debe retornar el código “jwt_007” con el detalle del error.



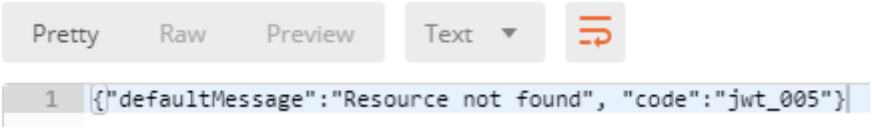
2.5. Si el token enviado se encuentra mal construido, el sistema debe enviar el código de error “jwt_008”, con el detalle del mismo.



2.6. Si el token no se puede desencriptar con el código guardado, el sistema debe enviar el código “jwt_009”, con el detalle del error.



2.7. Si el recurso solicitado no existe, el sistema lo debe informar mediante el código “jwt_005”.



2.8. Si el usuario no tiene permiso sobre el recurso solicitado, el sistema debe retornar el código “jwt_006”.

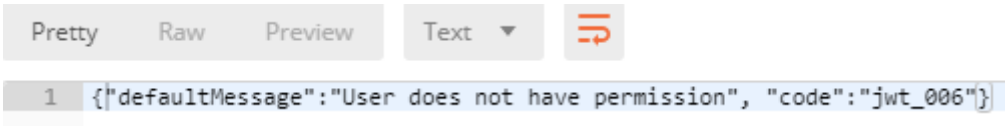
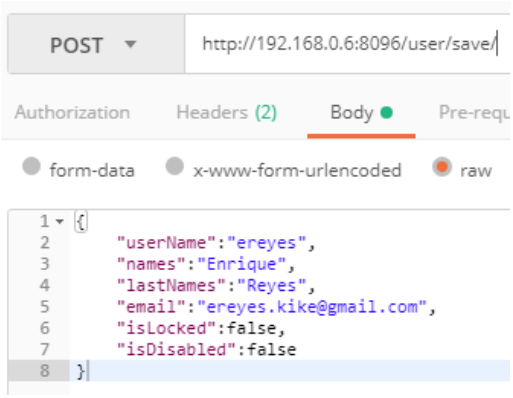


Tabla 26: Pruebas al caso de uso "Adicionar usuario"

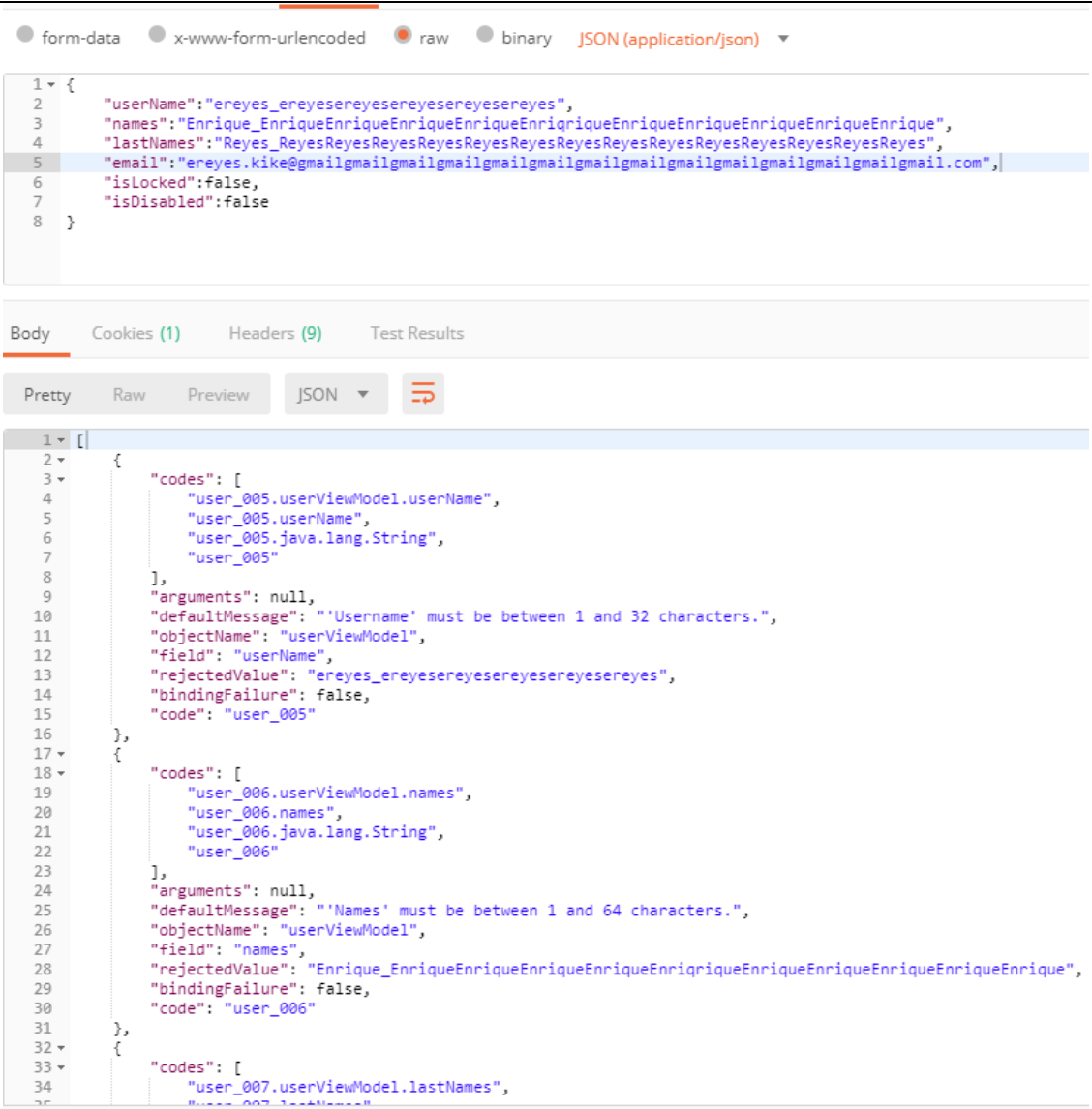
Información General	
Identificador de caso de uso:	CU-05
Nombre de caso de uso:	Adicionar usuario
Descripción Prueba:	El sistema debe permitir agregar un usuario con todos sus campos.
Responsable:	Equipo de pruebas
Prerrequisitos	
El usuario debe encontrarse autenticado y con permisos de agregar usuarios al sistema.	
Descripción de Casos de Prueba	
1- El sistema debe permitir agregar un usuario si todos los campos son ingresados correctamente. 2- El sistema debe validar todos los campos ingresados por el usuario. 3- El sistema debe verificar que no exista un usuario con el alias igual al alias del usuario que se desea agregar. 4- El sistema debe verificar que no exista un usuario con el correo electrónico igual al correo electrónico del usuario que se desea agregar. 5- El sistema debe comprobar sintácticamente el correo electrónico del usuario que se desea adicionar. 6- El sistema debe generar una contraseña aleatoria que será asignada al nuevo usuario y que además debe estar encriptada. 7- El sistema debe generar un log con los detalles del usuario agregado. 8- El sistema le debe enviar por correo electrónico la contraseña al nuevo usuario agregado.	
Servicio (s) Probados: `public ResponseEntity<?> save(@RequestBody @Valid UserViewModel user, BindingResult result)`	
Instrucciones de Prueba 1. Acceder al backend a la siguiente url: http://192.168.0.6:8096/user/save/ mediante algún programa que permita el envío de peticiones HTTP REST, por ejemplo, Postman.	

2. Se debe configurar además el método POST de acceso, agregar en la cabecera los datos del token enviado luego de la autenticación y los datos del usuario que se desea agregar en formato JSON.



2.1. Si los datos que se envía se encuentran correctos el sistema debe retornar en formato JSON los datos del usuario con el identificador generado y con estado 200 (OK).

2.2. Si algunos de los datos ingresados se encuentran erróneo el sistema debe retornar una lista de todos los errores con su descripción.



2.3. Si el alias o el correo electrónico ingresado se encuentra registrado en otro usuario el sistema debe retornar los detalles del error.

```
[
    {
        "codes": [
            "user_009.userViewModel.userName",
            "user_009.userName",
            "user_009.java.lang.String",
            "user_009"
        ],
        "arguments": null,
        "defaultMessage": "'Username' already added.",
        "objectName": "userViewModel",
        "field": "userName",
        "rejectedValue": "ereyes",
        "bindingFailure": false,
        "code": "user_009"
    },
    {
        "codes": [
            "user_010.userViewModel.email",
            "user_010.email",
            "user_010.java.lang.String",
            "user_010"
        ],
        "arguments": null,
        "defaultMessage": "'Email' already added.",
        "objectName": "userViewModel",
        "field": "email",
        "rejectedValue": "ereyes.kike@gmail.com",
        "bindingFailure": false,
        "code": "user_010"
    }
]
```


2.4. El sistema debe generar un log con los datos del usuario agregado.

```
: Looking up handler method for path /user/save/
: Returning handler method [public org.springframework.http.Resp
: Read [class ec.tesis.viewmodel.UserViewModel] as "application/
: Agregando usuario: ereyes|Enrique|Reyes|ereyes.kike@gmail.com
: Written [ec.tesis.viewmodel.UserViewModel@3ca97f37] as "applic
: Null ModelAndView returned to DispatcherServlet with name 'dis
: Successfully completed request
```

2.5. El sistema debe enviar por correo electrónico la contraseña generada.

5.4. Seguridad

La seguridad es un tema inherente al desarrollo de aplicaciones web. Un gran porciento de los atacantes persigue este tipo de aplicaciones, debido a la gran cantidad de información que se puede manejar y la delicadeza de la misma. Si un sistema informático comprometiera sus datos, podría provocar una reducción sustancial de sus clientes. Entre las diferentes vulnerabilidades de seguridad se encuentran los intentos

de autenticación y el control de acceso sin los permisos adecuados, los ataques de inyección de datos a través de SQL Injection y los ataques de inyección de contenido mediante Cross-Site-Scripting. (La importancia de la seguridad en las aplicaciones web., n.d.)

El aplicativo desarrollado cuenta con varios mecanismos que fortalecen su seguridad, entre los cuales se pueden mencionar:

- ✓ El acceso a datos se realiza a través de las interfaces que provee la clase JpaRepository, perteneciente al framework de persistencia Hibernate. En ningún momento se acceden a los datos mediante consultas no parametrizadas.

```java
@Repository
public interface PermissionRepository extends JpaRepository<PermissionModel, UUID> {

    List<PermissionModel> findAllByRole(RoleModel role);

    List<PermissionModel> findAllByResource(ResourceModel resource);

    Optional<PermissionModel> findByRoleInAndResource(List<RoleModel> roles, ResourceModel resource);

    List<PermissionModel> findAllByRoleIn(List<RoleModel> roles);

    @Transactional
    void deleteByRole(RoleModel role);

    @Transactional
    void deleteByResource(ResourceModel resource);

}
```

- ✓ Correcta configuración del mapeo de los controladores y acceso a sus diferentes métodos a través de las anotaciones @RequestMapping y sus derivados @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, entre otras.

```java
@RestController
@RequestMapping("/company")
public class CompanyController {

    @Autowired
    private CompanyService service;

    @PostMapping(value = "/save/")
    public ResponseEntity<?> save(@RequestBody @Valid CompanyViewModel company,
                                  BindingResult result) throws Exception {
        try {
            CompanyModel response = service.save(Mapper.ConvertToCompanyModel(company),
                                                 result);

            if (result.hasErrors())
                return new ResponseEntity<>(result.getAllErrors(),
                                            HttpStatus.UNPROCESSABLE_ENTITY);

            return new ResponseEntity<>(Mapper.ConvertToCompanyViewModel(response),
                                        HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(e, HttpStatus.BAD_REQUEST);
        }
    }
```

- ✓ Validaciones de las entradas de datos mediante las clases de tipo <Validator> correspondiente a cada una de las entidades definidas.

```java
@Component
public class UserValidator extends BaseValidator implements Validator {

    @Autowired
    UserRepository repository;

    public boolean supports(Class<?> clazz) {
        return UserModel.class.isAssignableFrom(clazz);
    }

    public void validate(Object target, Errors errors)
    {
        ValidationUtils.rejectIfEmpty(errors, "userName", "user_001", "'Username' should not be empty.");
        ValidationUtils.rejectIfEmpty(errors, "names", "user_002", "'Names' should not be empty.");
        ValidationUtils.rejectIfEmpty(errors, "lastNames", "user_003", "'Lastnames' should not be empty.");
        ValidationUtils.rejectIfEmpty(errors, "email", "user_004", "'Email' should not be empty.");

        UserModel user = (UserModel) target;

        if (user.getUserName().length() > 32)
            errors.rejectValue("userName", "user_005", "'Username' must be between 1 and 32 characters.");

        if (user.getNames().length() > 64)
            errors.rejectValue("names", "user_006", "'Names' must be between 1 and 64 characters.");

        if (user.getLastNames().length() > 64)
            errors.rejectValue("lastNames", "user_007", "'Lastnames' must be between 1 and 64 characters.");

        if (user.getEmail().length() > 64)
            errors.rejectValue("email", "user_008", "'Email' must be between 1 and 64 characters.");

        Optional<UserModel> userAux = repository.findByUserName(user.getUserName());

        // Verificando que no existe otro usuario con el mismo username.
        if (userAux.isPresent() && !userAux.get().getId().equals(user.getId()))
            errors.rejectValue("userName", "user_009", "'Username' already added.");

        userAux = repository.findByEmail(user.getEmail());

        // Verificando que no existe otro usuario con el mismo email.
        if (userAux.isPresent() && !userAux.get().getId().equals(user.getId()))
            errors.rejectValue("email", "user_010", "'Email' already added.");

        if (!validate(EMAIL_REGEX, user.getEmail()))
            errors.rejectValue("email", "user_011", "'Email' not valid.");
    }

}
```

- ✓ Uso del framework Spring Security para la configuración de la autenticación y el control de acceso.

```java
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors()
        .and()
        .csrf()
        .disable()
        .authorizeRequests()
        // Allow anonymous visitors to auth controller methods
        .antMatchers("/login","/about").permitAll()
        // All other request need to be authenticated
        .anyRequest().authenticated()
        .and()
        .addFilterBefore(new LoginFilter("/login",
                         authenticationManager()),
                         UsernamePasswordAuthenticationFilter.class)
        .addFilterBefore(new UpdatePasswordFilter("/update-password",
                                                  authenticationManager()),
                                                  UsernamePasswordAuthenticationFilter.class)
        .addFilterBefore(new JwtFilter(), UsernamePasswordAuthenticationFilter.class);
}
```

```
@Override
public Authentication attemptAuthentication(HttpServletRequest req, HttpServletResponse res)
        throws IOException {
    SecurityUser user = null;

    try {
        user = new ObjectMapper().readValue(req.getReader(), SecurityUser.class);

        return getAuthenticationManager().authenticate(
                new UsernamePasswordAuthenticationToken(
                        user.getUserName(), user.getPassword(), Collections.emptyList()));
    } catch(BadCredentialsException e) {
        SecurityUtils.addHeader(res, "", -1);
        res.getWriter().print("{\"defaultMessage\":\"Credentials are invalid\", \"code\":\"login_001\"}");
        res.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        return null;
    } catch(DisabledException e) {
        SecurityUtils.addHeader(res, "", -1);
        res.getWriter().print("{\"defaultMessage\":\"Account is disabled\", \"code\":\"login_002\"}");
        res.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        return null;
    } catch(LockedException e) {
        SecurityUtils.addHeader(res, "", -1);
        res.getWriter().print("{\"defaultMessage\":\"Account is locked\", \"code\":\"login_003\"}");
        res.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        return null;
    } catch(CredentialsExpiredException e) {
```

- ✓ Generación de un token válido luego de una satisfactoria autenticación, el cual será verificado en cada una las solicitudes realizadas al sistema.

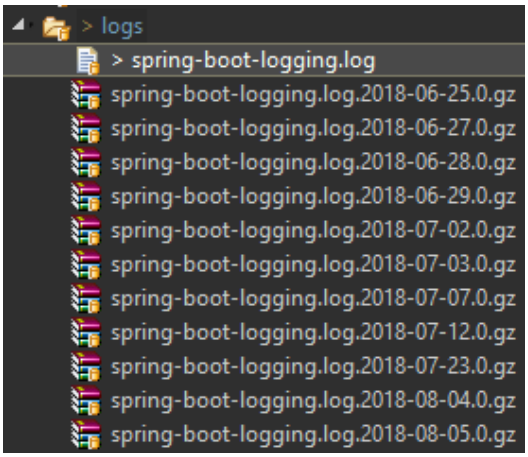
```
@Override
protected void successfulAuthentication(HttpServletRequest req,
                                        HttpServletResponse res,
                                        FilterChain chain,
                                        Authentication auth) throws IOException {
    try {
        int minutes = Integer.parseInt(AppService.getProperty("tesis.security.time"));
        SecurityUtils.addHeader(res, auth.getName(), minutes);

        ServletContext servletContext = req.getServletContext();
        WebApplicationContext webApplicationContext = WebApplicationContextUtils.getWebApplicationContext(servletContext);
```

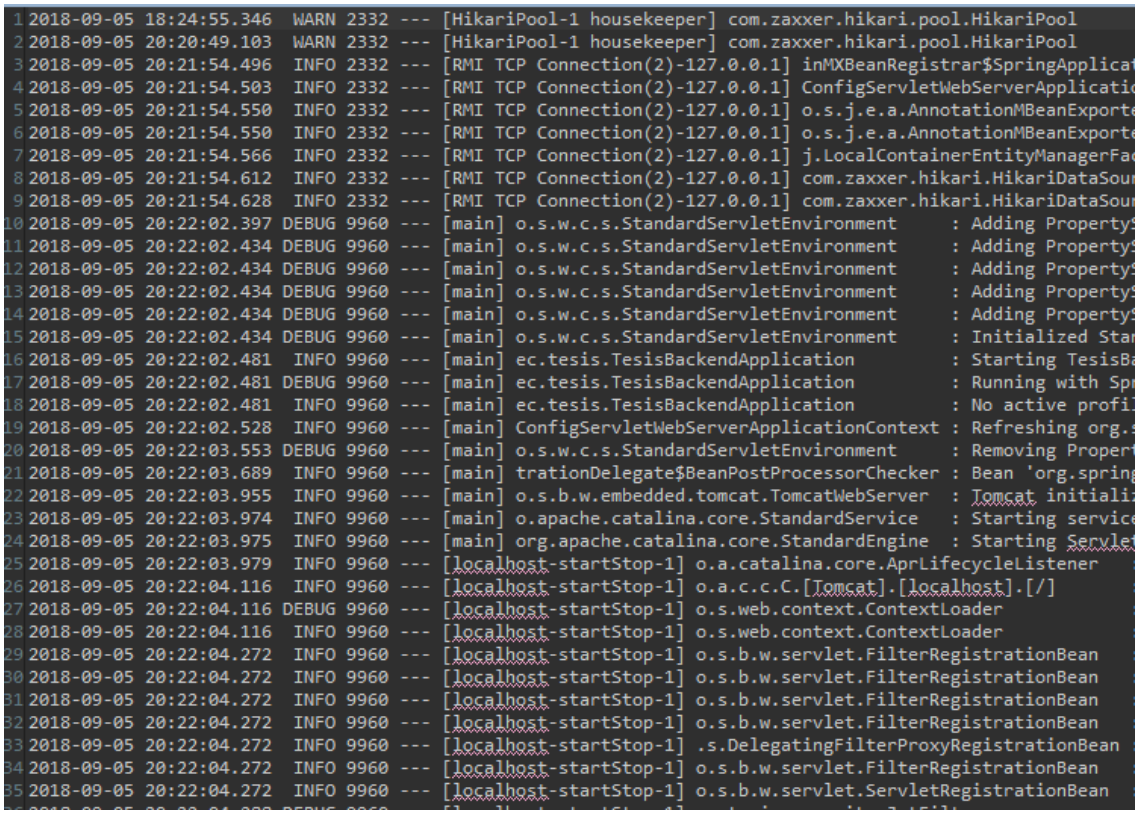
```
@Component
public class JwtFilter extends GenericFilterBean {

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
            throws IOException, ServletException {
        try {
            String token = ((HttpServletRequest) req).getHeader("Tesis");
            String base64EncodedKeyBytes = AppService.getProperty("tesis.security.secret");

            if (token != null) {
                String subject = Jwts.parser()
                                     .setSigningKey(base64EncodedKeyBytes)
                                     .parseClaimsJws(token.replace("Init:", ""))
                                     .getBody()
                                     .getSubject();

                ServletContext servletContext = req.getServletContext();
                WebApplicationContext webApplicationContext = WebApplicationContextUtils.getWebApplicationContext(servletContext);

                UserService userService = webApplicationContext.getBean(UserService.class);
                UserDetails user = userService.loadUserByUsername(subject);

                if (!user.isEnabled())
                    throw new DisabledException("Account is disabled");

                if (!user.isAccountNonLocked())
                    throw new LockedException("Account is locked");

                if (!user.isCredentialsNonExpired())
                    throw new CredentialsExpiredException("Account's credentials have expired");
```

- ✓ Validación del acceso a los recursos solicitados.

```
String path = ((HttpServletRequest) req).getServletPath();
int endIndex = path.indexOf('/', path.indexOf('/', path.indexOf('/') + 1) + 1);

ResourceService resourceService = webApplicationContext.getBean(ResourceService.class);
List<ResourceModel> resources = resourceService.findByPath(path.substring(0, endIndex + 1));

if (resources.size() == 0)
    throw new ResourceClosedException("Resource not found");

if (!subject.equals("root")) {
    ProfileService profileService = webApplicationContext.getBean(ProfileService.class);
    List<ProfileModel> profiles = profileService.findAllByUserName(subject);

    List<RoleModel> roles = profiles.stream().map(p -> p.getRole()).collect(Collectors.toList());

    PermissionService permissionService = webApplicationContext.getBean(PermissionService.class);

    if (!permissionService.rolesHasPermission(roles, resources.get(0)))
        throw new ResourceAccessException("User does not have permission");
}

Authentication authentication = subject != null ? new UsernamePasswordAuthenticationToken(subject,
SecurityContextHolder.getContext().setAuthentication(authentication);

((HttpServletRequest)req).setAttribute("subject", subject);
chain.doFilter(req, res);
```

- ✓ Generación de logs que almacenan las acciones que ejecuta el sistema y permiten ser auditados.





- ✓ Administración de la base de datos Postgres:
  - o Configuración de los usuarios que pueden acceder y su tipo de autenticación.
  - o Definición del IP o subred desde donde se puede acceder a la base de datos.
  - o Parametrización del número máximo de clientes conectados a la vez.
  - o Administración de roles.
  - o Definición de vistas para proteger las tablas.
  - o Creación de certificados digitales para la autenticación.

5.5. Aplicación

El acceso al aplicativo se realiza a través de la autenticación mediante usuario y contraseña. No existe función alguna que permita su ejecución sin una previa identificación del usuario.

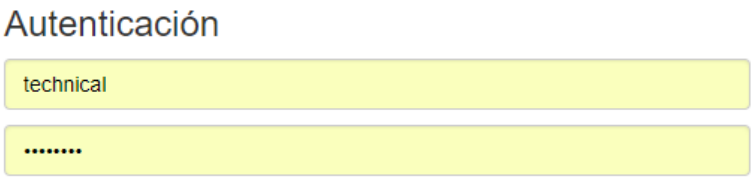


Ilustración 27: Pantalla de autenticación

Existe una vista para los usuarios y roles, donde se pueden listar, agregar, eliminar y editar cada uno de ellos. En el caso de los usuarios también permite las acciones de bloqueo y reseteo de las contraseñas.

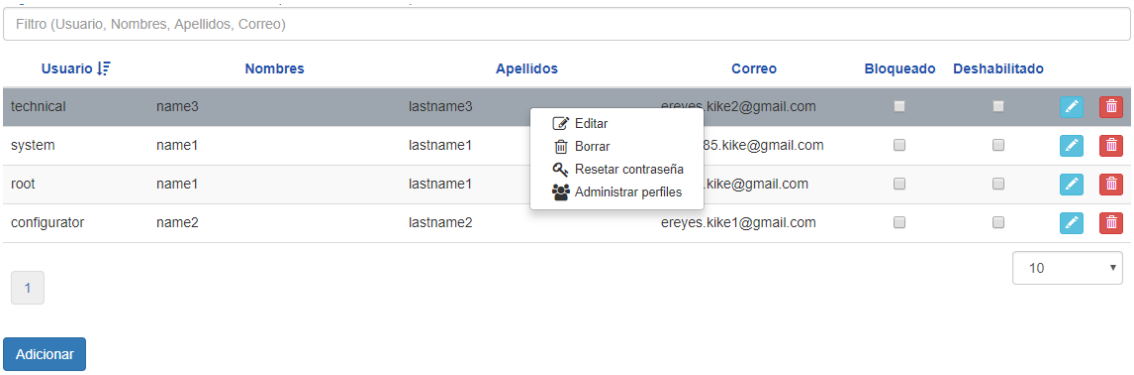


Ilustración 28: Pantalla de usuarios registrados

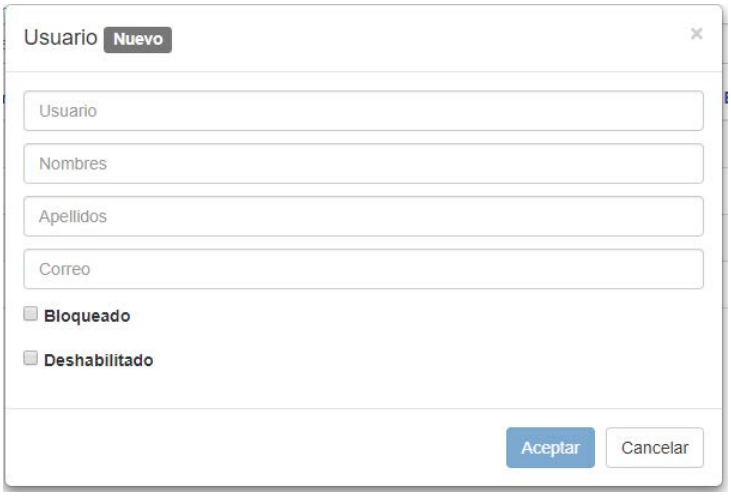


Ilustración 29: Pantalla de registro de usuarios

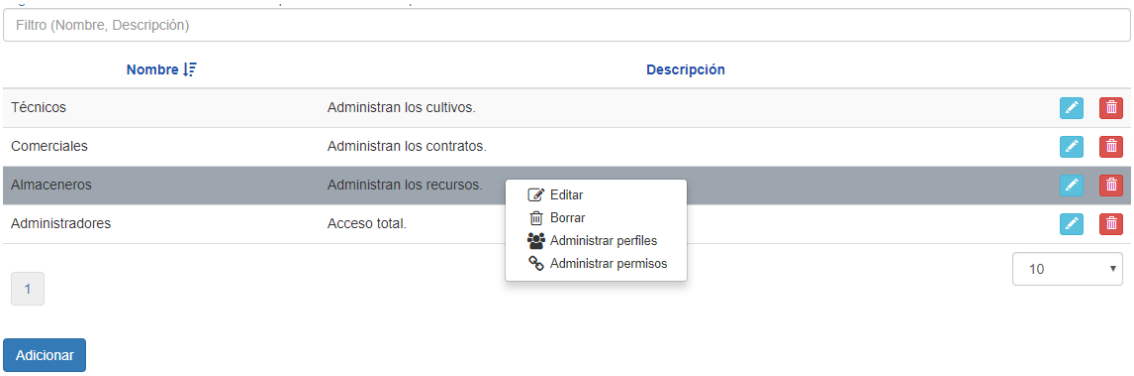


Ilustración 30: Pantalla de roles registrados

Ilustración 31: Pantalla de registro de roles

El sistema permite configurar diferentes roles con el objetivo de agrupar los usuarios registrados, de tal forma que un usuario puede pertenecer a uno o varios roles a la vez, con la ventaja que puede acceder a todos los recursos configurados en cada uno de los roles.

Ilustración 32: Pantalla de roles asociados por usuario

La aplicación cuenta con varios recursos asociados, que no son más que las diferentes rutas a las que se acceden al backend, donde las rutas son conocidas como “actions” y las clases donde se implementas son llamadas “controllers”. Mediante una vista se pueden relacionar los diferentes recursos a los roles. El objetivo de la configuración del control de acceso es obtener una relación entre los usuarios y los recursos existentes en el aplicativo, verificando dicha relación en cada una de las peticiones realizadas por el frontend.

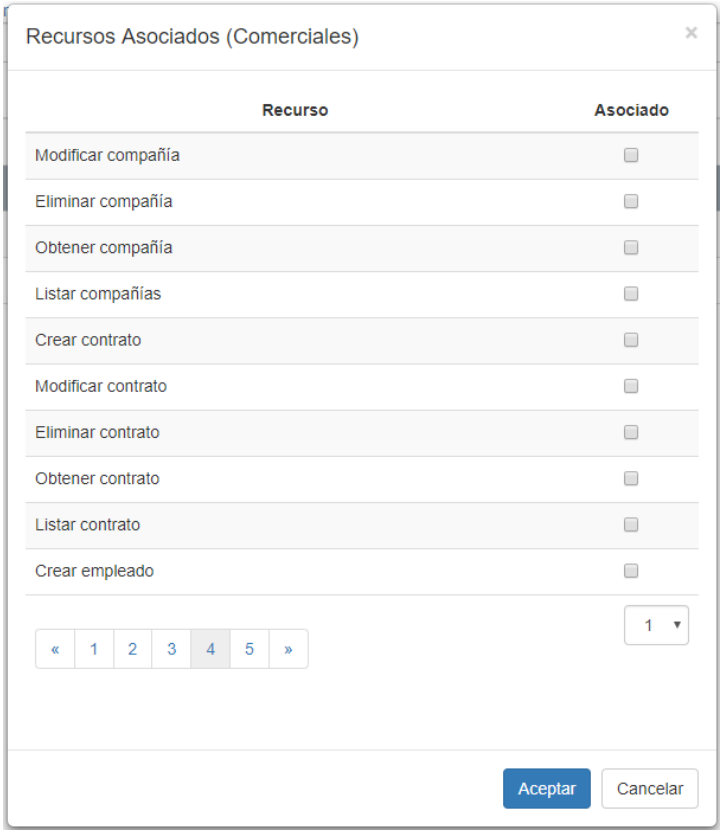


Ilustración 33: Pantalla de recursos asociados por rol

Explicada la parte relacionada a la seguridad, le siguen las pantallas relacionadas al negocio.

Cada una de las siguientes tablas, y las antes mencionadas, cuentan con las funcionalidades de filtrar, ordenar en cada una de las columnas, paginado y selección de números de elementos por páginas, facilitándoles al usuario una cómoda interacción con la interfaz visual presentada.

Primeramente, se encuentra la vista para gestionar los cultivos, formados por su nombre y descripción.

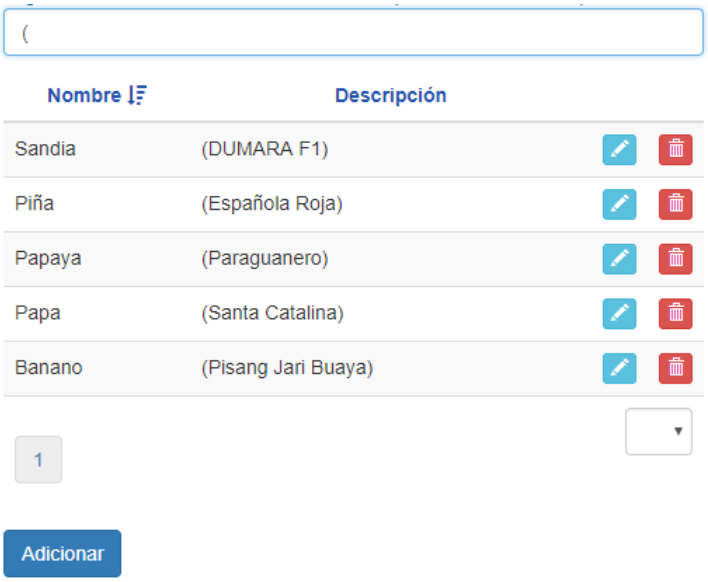


Ilustración 34: Pantalla de cultivos registrados

Ilustración 35: Pantalla de registro de cultivos

En la siguiente pantalla se pueden gestionar los diferentes productos necesarios para llevar a cabo la propagación de los cultivos anteriormente seleccionados. Los productos están formados por su nombre, descripción, tipo de unidad de medida, la cantidad que se encuentra en existencia y su unidad de medida.

Ilustración 36: Pantalla de productos registrados

Ilustración 37: Pantalla de registro de productos

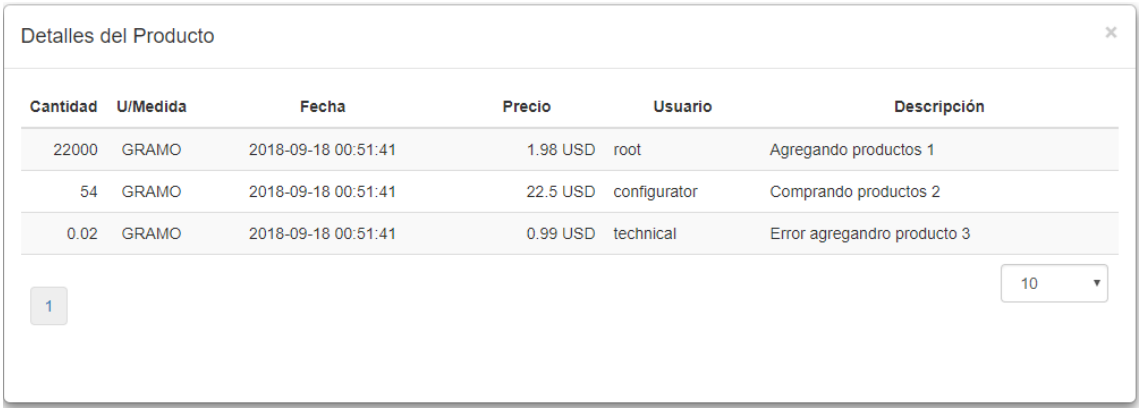


Ilustración 38: Pantalla de detalles de productos registrados

El sistema facilita la gestión de las empresas que participan en los contratos, registrándoles sus nombres, ruc, representante legal, teléfono y ubicación.

Ilustración 39: Pantalla de empresas registradas

Ilustración 40: Pantalla de registro de empresas

A través de la siguiente vista el sistema permite registrar los empleados que pueden ser seleccionados en los contratos, detallándose sus nombre, apellidos y correo electrónico.

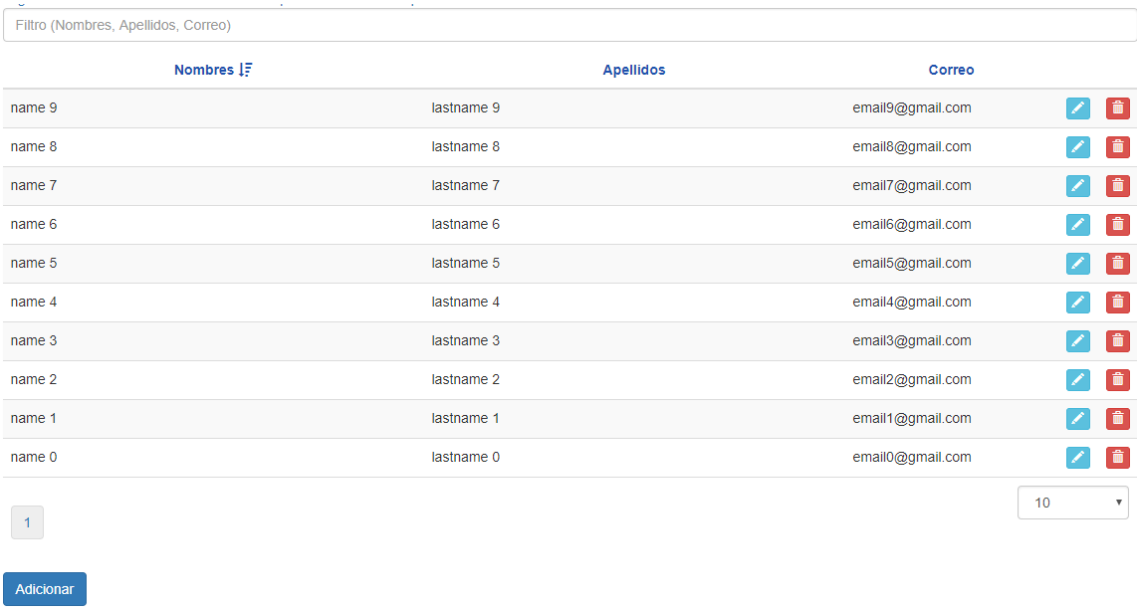


Ilustración 41: Pantalla de empleados registrados

Ilustración 42: Pantalla de registro de empleados

Uno de los principales objetivos del aplicativo es realizar la administración de los contratos, utilizándose los cultivos, productos, empresas y empleados configurados con anterioridad. En la siguiente pantalla se pueden seleccionar cada uno de los elementos implicados en un contrato, con las siguientes particularidades:

- La fecha de fin del contrato debe ser mayor que la fecha de inicio.
- Los productos y las fechas de entregas configuradas deben ser únicos para cada cultivo seleccionado.
- Los empleados seleccionados participarán en el desarrollo de todos los productos del contrato.
- Al seleccionar el cultivo y el producto, se debe ingresar además la cantidad.

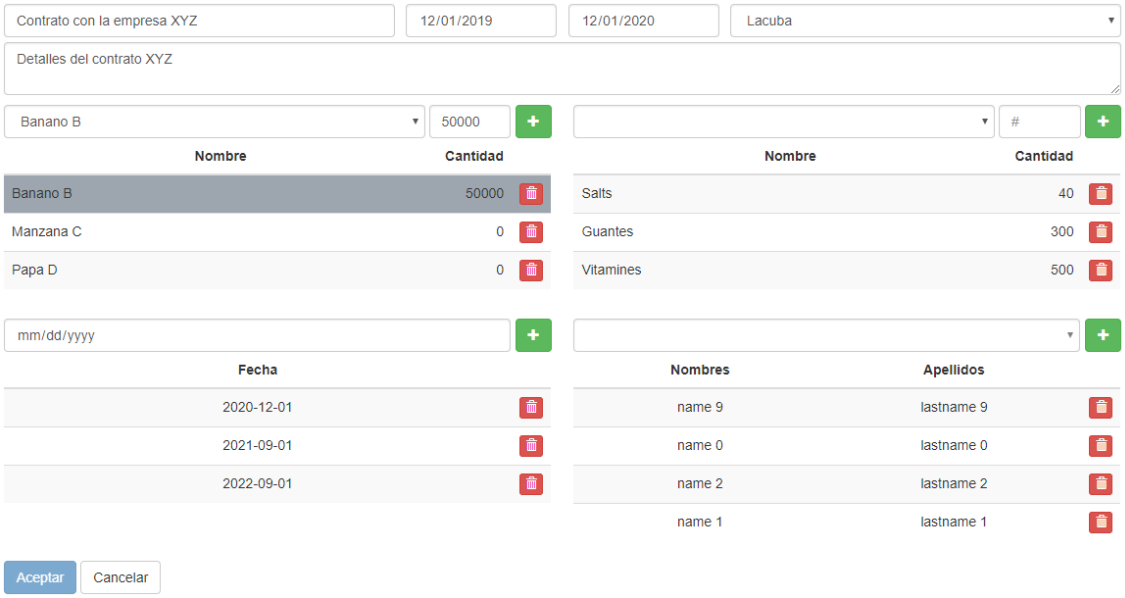


Ilustración 43: Pantalla de registro de contratos

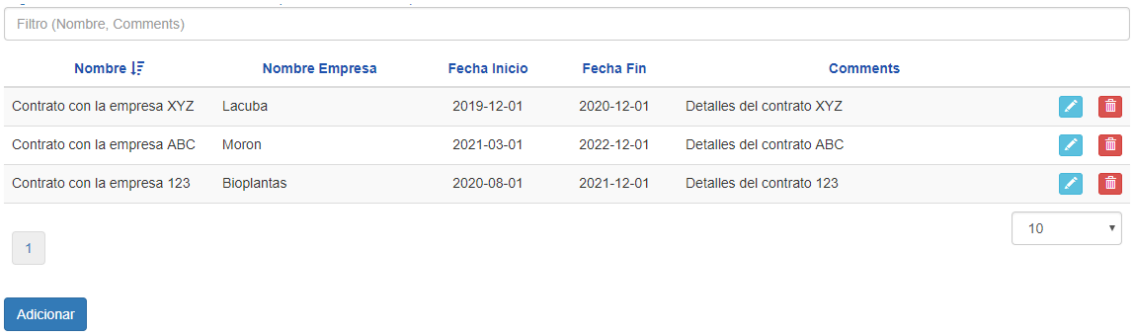


Ilustración 44: Pantalla de contratos registrados

A partir de las pantallas configuradas, el sistema permite obtener varios reportes que le permite al usuario tener una mejor visión del desarrollo del proyecto. Entre los diferentes reportes se encuentran:

- Un diagrama de Gantt que visualiza los contratos actuales.
- Un diagrama de Gantt con las fechas de los entregables.
- Gráfico que visualice las empresas que participan en los contratos registrados.
- Gráfico que muestra los cultivos registrados.
- Gráfico que representa la cantidad de productos existentes.

El sistema cuenta con las siguientes funcionalidades adicionales:

- Cada día envía un mensaje por correo electrónico a los usuarios con permiso para gestionar los contratos, recordándoles las fechas de entrega.
- Mensaje por correo electrónico a los usuarios con permiso para registrar los productos, con el objetivo de advertirles antes de que se agoten los mismos.

6. Conclusiones y trabajo futuro

6.1. Conclusiones generales

A partir de la investigación realizada para la elaboración del producto presentado, utilizando a RUP como metodología idónea para lograr una mejor comprensión de los objetivos planteados inicialmente, se arriba a las siguientes conclusiones:

- Se definieron los requerimientos que garantizaron la operatividad del aplicativo desarrollado.
- Se desarrolló una aplicación web que permite la gestión de proyectos de micropropagación de plantas biotecnológicas teniendo en cuenta las premisas del software libre.
- Se validó el sistema a través de diferentes pruebas funcionales de caja negra.
- Se evaluó el aplicativo desarrollado por varios especialistas en el tema abordado.

7. Recomendaciones

Luego de culminado el desarrollo de la aplicación se realizan las siguientes recomendaciones:

- Desarrollar una aplicación móvil que brinde las funcionalidades claves implementadas en el ambiente web.
- Agregar otros métodos de autenticación que puedan fortalecer el acceso al sistema permitiendo una doble autenticación.
- Integrar algoritmos de inteligencia artificial al sistema que permita la predicción de nuevas configuraciones que maximicen la productividad del cliente.
- Mantener el sistema actualizado con las nuevas tendencias de frameworks y bibliotecas de desarrollo.

8. Bibliografía

Agramonte, D., Orellana, P., Suárez-Castella, M., Jiménez, M. A., Santana Aguilar, I., Jiménez, E., . . . Rodríguez, S. (2006). Biofábricas para la micropropagación de especies vegetales. Obtenido de https://revista.ibp.co.cu/index.php/BV/article/view/477/863

Alagumani, T. (2005). Economic Analysis of Tissue-cultured Banana and Sucker-propagated Banana. Obtenido de http://ageconsearch.umn.edu/bitstream/58413/2/T-Alagumai.pdf

Bhatia, S., & Bera, T. (2015).

Chen, C. (2016). Cost analysis of plant micropropagation of Phalaenopsis. Obtenido de https://www.researchgate.net/publication/299656123_Cost_analysis_of_plant_micropropagation_of_Phalaenopsis

Dayley, B., Dayley, B., & Dayley, C. (2018). *Learning Angular: A Hands-On Guide to Angular 2 and Angular 4.* Obtenido de https://books.google.com.ec/books?id=BHs2DwAAQBAJ&printsec=frontcover&dq=angular+2+pdf&hl=es-419&sa=X&ved=0ahUKEwjvgKTt8cfbAhUN7VMKHQrBD2MQ6AEIbjAJ#v=onepage&q&f=false

Development, G. T. (2018). *WHAT IS POSTGRESQL?* Obtenido de postgresql.org: https://www.postgresql.org/about/

Eguíluz Pérez, J. (2009). *Introducción a JavaScript.*

Foundation, T. A. (s.f.). *Apache Tomcat*®. Obtenido de http://tomcat.apache.org/

Fowler, M., & Scott, K. (1997). *UML gota a gota.* Obtenido de https://ingenieriasoftware2011.files.wordpress.com/2011/07/uml-gota-a-gota.pdf

Framework para el desarrollo ágil de aplicaciones. (s.f.). Obtenido de https://www.acens.com/wp-content/images/2014/03/frameworks-white-paper-acens-.pdf

Galvão Mendonça, E., Cristina Stein, V., Henrique de Carvalho, H., Régis Santos, B., Alberto Beijo, L., & Vilela Paiva, L. (2016). The use of continuous, Temporary Immersion Bioreactor system and semisolid culture medium for the production of Eucalyptus camaldulensis clones.

Gérard, S. (s.f.). *Papyrus 2.0: What, Why and What's Next*. Obtenido de https://www.eclipse.org/community/eclipse_newsletter/2016/april/article1.php

Gutierrez, D. (2010). *Frameworks y Componentes.* Obtenido de http://www.codecompiling.net/files/slides/IS_clase_10_frameworks_components.pdf

Gutiérrez, J. J. (s.f.). Obtenido de ¿Qué es un framework web? : http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Hempel, M., Хемпел, М., & Хемпел, М. (1986). Some Economical Aspects of Commercial Micropropagation. Obtenido de https://www.tandfonline.com/doi/pdf/10.1080/02052067.1986.10824247

Hofmann, M., Schnabel, E., & Stanley, K. (2016). Microservices Best Practices for Java. Obtenido de https://www.redbooks.ibm.com/redbooks/pdfs/sg248357.pdf

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software.* Obtenido de https://es.slideshare.net/gagamonsterbadromance/el-proceso-unificado-de-desarrollo-de-software-jacobson-booch-rumbaugh

Jacobson, I., Spence, I., & Bittner, K. (2013). CASOS DE USO 2.0: La guía para ser exitoso con los casos de uso. Obtenido de https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use_case_2.0_-_spanish_translation.pdf

Javascript language. (2015). Obtenido de https://www.tutorialspoint.com/javascript/javascript_tutorial.pdf

Jiménez Marín, A., & Pérez Montes, F. M. (2016). *Aprende a programar con Java.* Obtenido de https://books.google.com.ec/books?id=is3BCwAAQBAJ&printsec=frontcover&dq=Jim%C3%A9nez+Mar%C3%ADn,+Alfonso;+P%C3%A9rez+Montes,+Francisco+Manuel&hl=es-419&sa=X&ved=0ahUKEwic7Jml38HbAhVC2lMKHRx5CigQ6AEIJjAA#v=onepage&q=Jim%C3%A9nez%20Mar%C3%ADn%2C%20Alfonso%

Jovanović, Ž., Jagodić, D., Vujičić, D., & Ranđić, S. (2017). JAVA SPRING BOOT REST WEB SERVICE INTEGRATION WITH JAVA ARTIFICIAL INTELLIGENCE WEKA FRAMEWORK. Obtenido de https://www.researchgate.net/publication/321757987_Java_Spring_Boot_Rest_WEB_Service_Integration_with_Java_Artificial_Intellgence_Weka_Framework

Kadam, D., Chhatre, A., Lavale, S., & Shinde, N. (2018). Low-cost alternatives for conventional tissue culture media. Obtenido de https://www.ijcmas.com/7-4-2018/Dipak%20D.%20Kadam,%20et%20al.pdf

Kahlert, T., & Giza, K. (2016). *Visual Studio Code Tips & Tricks Vol. 1.* Obtenido de http://download.microsoft.com/download/8/A/4/8A48E46A-C355-4E5C-8417-E6ACD8A207D4/VisualStudioCode-TipsAndTricks-Vol.1.pdf

La importancia de la seguridad en las aplicaciones web. (s.f.). Obtenido de https://blog.nerion.es/la-importancia-de-la-seguridad-en-las-aplicaciones-web-como-crear-aplicaciones-web-seguras/

Larman, C. (2003). *UML y Patrones.* Obtenido de http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf

Microsoft. (2018). *Documentation for Visual Studio Code*. Obtenido de code.visualstudio.com: https://code.visualstudio.com/docs

Mohedano, J., Saiz, J. M., & Salazar Román, P. (2012). *Iniciación a javascript.* Obtenido de https://bv.unir.net:2056/lib/univunirsp/reader.action?docID=3214795&query=&pg=11

Navarro Marset, R. (2007). *REST vs Web Services.* Obtenido de http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

Ordax Cassá, J. M., & Ocaña Díaz-Ufano, P. A. (s.f.). *Programación web en Java.* Obtenido de https://zb9vk6fg2q.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info%3Asid%2Fsummon.serialssolutions.com&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=book&rft.title=Programacio%CC%81n+web+en+java&rf

Orellana, P., Suárez-Castellá, M., Triana, R., Sarría, Z., Pons, M., León, M., . . . Pérez, Z. (2008).

que-es-typescript. (s.f.). Obtenido de devcode.la: https://devcode.la/blog/que-es-typescript/

Rangel-Estrada, S., Hernández-Meneses, E., & Hernández-Arenas, M. (2016).

Rumbaugh, J., Jacobson, I., & Booch, G. (2000). *El lenguaje unificado de modelado. Manual de referencia.* Obtenido de https://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf

Sahu, J., & Kumar Sahu, R. (2013). A Review on Low Cost Methods for In Vitro Micropropagation of Plant Through Tissue Culture Technique. Obtenido de http://www.ukjpb.com/article_details.php?id=7

Sánchez Asenjo, j. (2009). *Programación básica en Lenguaje Java.* Obtenido de https://openlibra.com/es/book/download/programacion-basica-en-java

Solano, J. (2011). Obtenido de Lenguajes de Programación: https://compinformatidf.files.wordpress.com/2012/03/cap7-lengprogram-cc101.pdf

Sommerville. (2005). *Ingeniería del software* (Séptima ed.). Obtenido de http://zeus.inf.ucv.cl/~bcrawford/EnfoquesDeDesarrolloDeSwYLenguajesDeModelado/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Torres Remon, M. (2013). *Desarrollo de aplicaciones con JAVA.* Obtenido de https://books.google.com.ec/books?id=gwkvDgAAQBAJ&printsec=frontcover&dq=que+es+java+pdf&hl=es-419&sa=X&ved=0ahUKEwjF76DZ1sDbAhXrqFkKHeyjAjcQ6AEIODAD#v=onepage&q&f=false

TypeScript. (2013). Obtenido de https://openlibra.com/es/book/download/typescript-language-specification

typescript-javascript. (s.f.). Obtenido de arsys.es: https://www.arsys.es/blog/programacion/typescript-javascript/

UNIR. (2018). Computación en el Servidor Web (ISW) - PER5 2017-2018 - Tema 6.

UNIR. (2018). Metodologías, Desarrollo y Calidad de la Ingeniería de Software (ISW) - PER5 2017-2018.

UNIR. (2018). Metodologías, Desarrollo y Calidad de la Ingeniería de Software (ISW) - PER5 2017-2018 - Tema 2.

UNIR. (2018). Plataformas de Desarrollo de Software (ISW) - PER5 2017-2018 - Tema 1.

UNIR. (2018). Plataformas de Desarrollo de Software (ISW) - PER5 2017-2018 - Tema 2.

Vukotic, A., & Goodwill, J. (2011). *Apache Tomcat 7.* Obtenido de https://www.gocit.vn/files/Apress.Apache.Tomcat.7-www.gocit.vn.pdf

9. Anexos

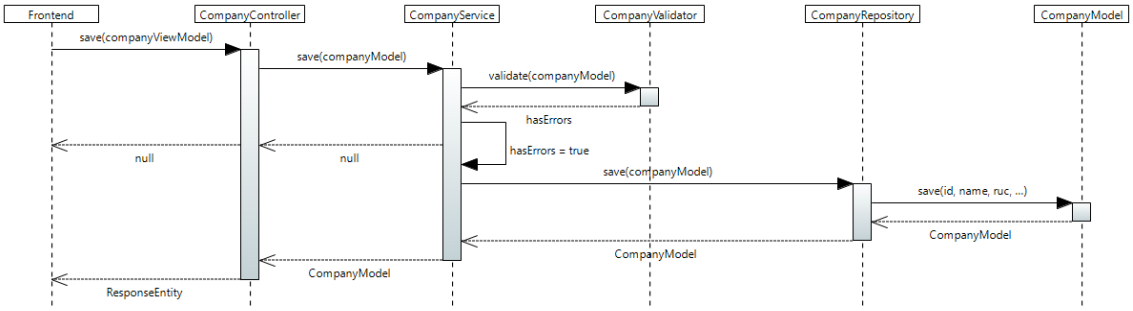


Ilustración 45: Diagrama de secuencia para agregar empresas

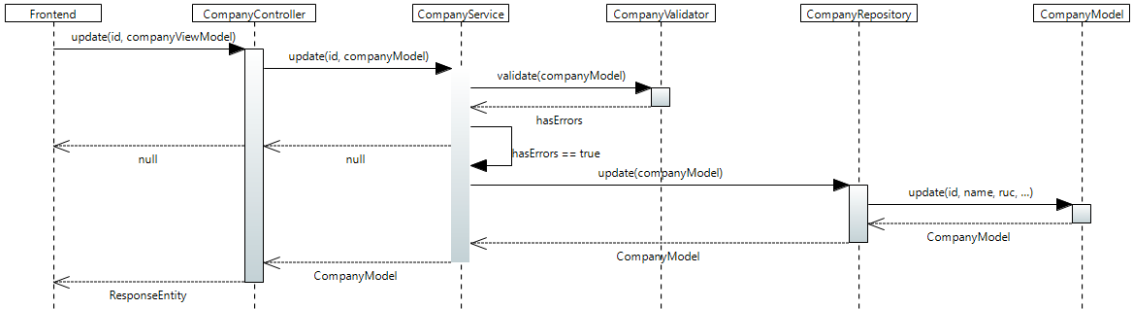


Ilustración 46: Diagrama de secuencia para editar empresas

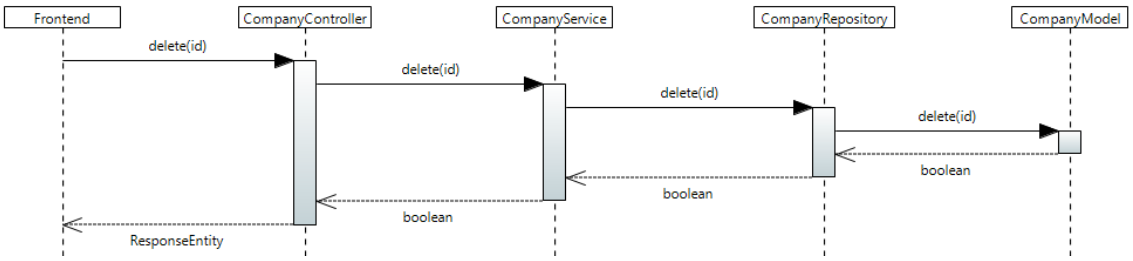


Ilustración 47: Diagrama de secuencia para eliminar empresas

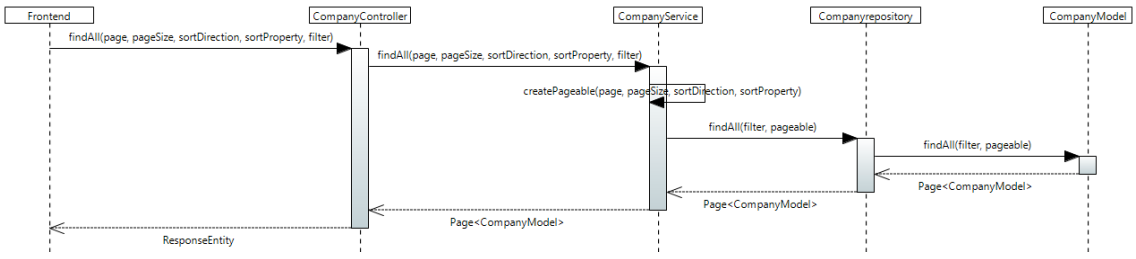


Ilustración 48: Diagrama de secuencia para listar empresas

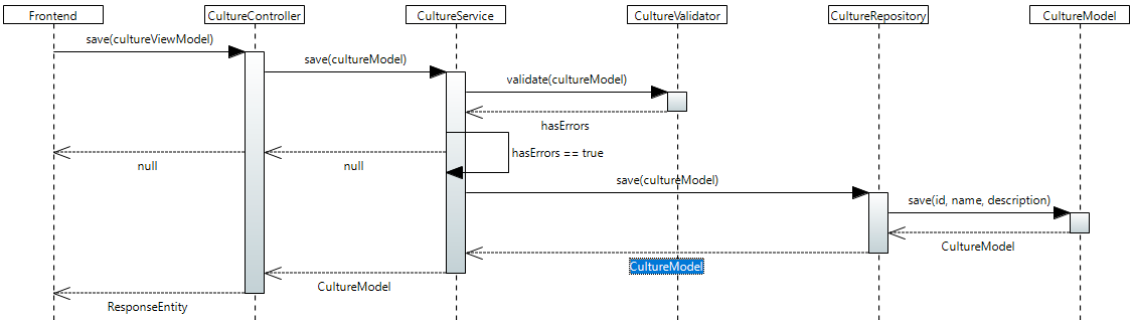


Ilustración 49: Diagrama de secuencia para agregar cultivos

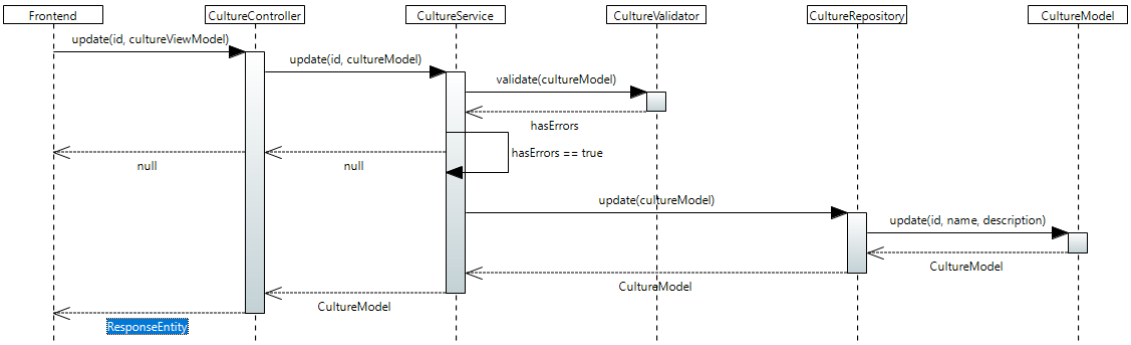


Ilustración 50: Diagrama de secuencia para editar cultivos

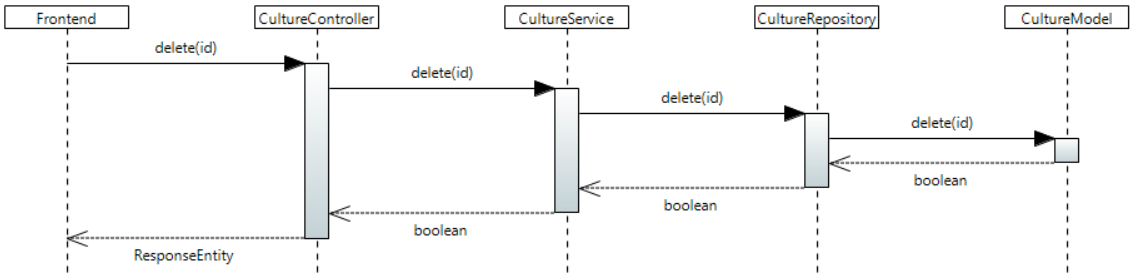


Ilustración 51: Diagrama de secuencia para eliminar cultivos

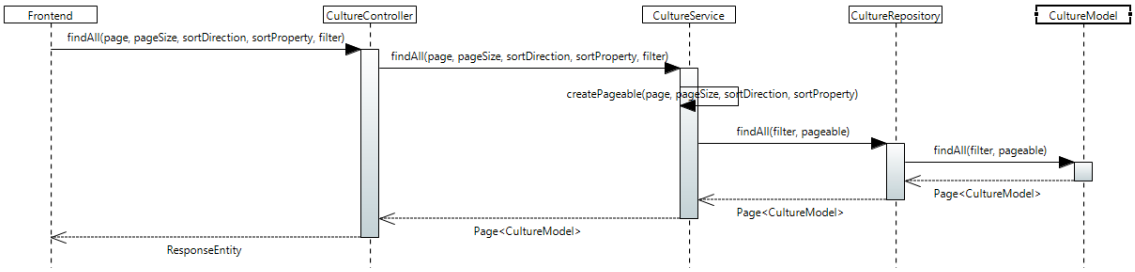


Ilustración 52: Diagrama de secuencia para listar cultivos

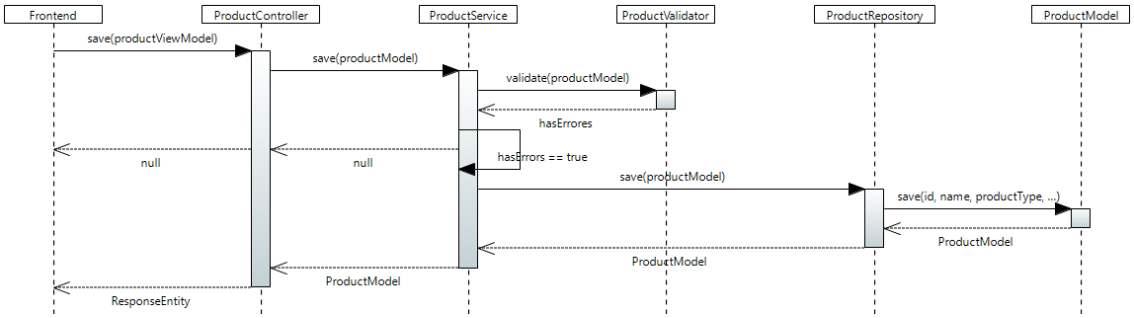


Ilustración 53: Diagrama de secuencia para agregar productos

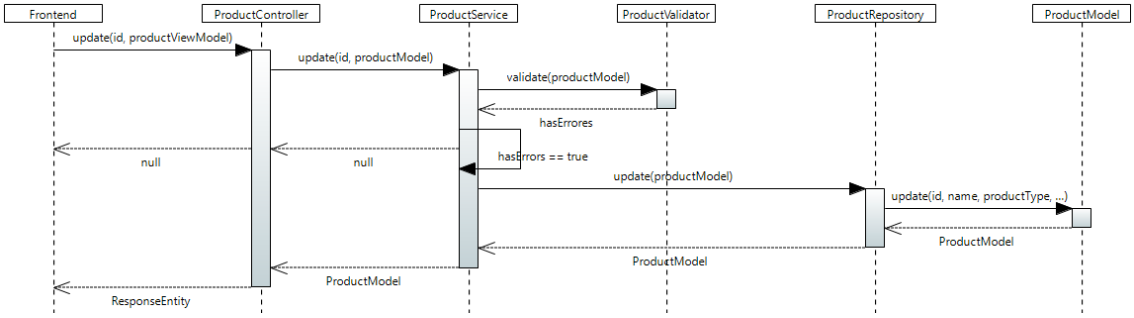


Ilustración 54: Diagrama de secuencia para editar productos

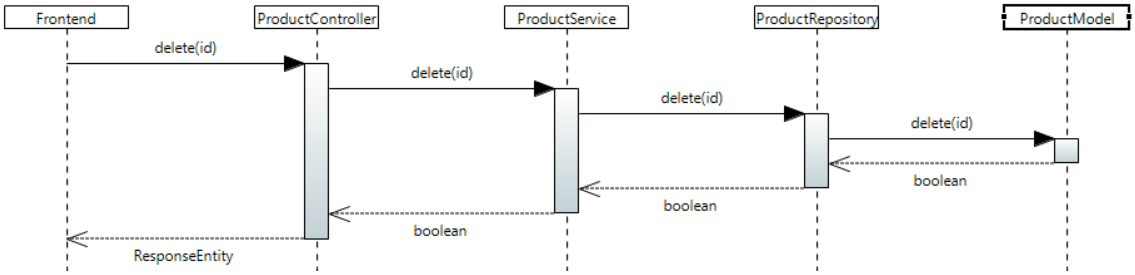


Ilustración 55: Diagrama de secuencia para eliminar productos

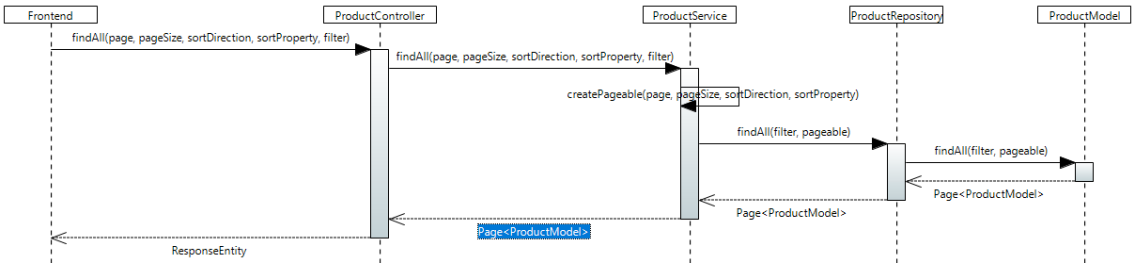


Ilustración 56: Diagrama de secuencia para listar productos

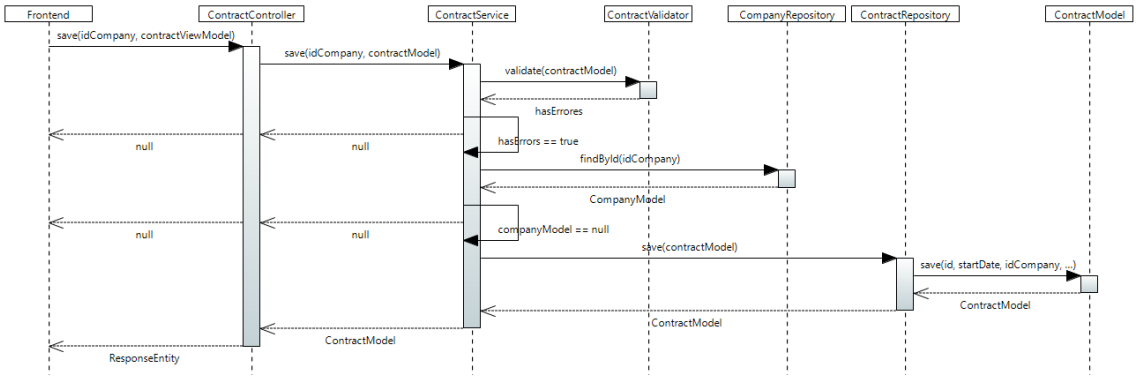


Ilustración 57: Diagrama de secuencia para agregar contratos

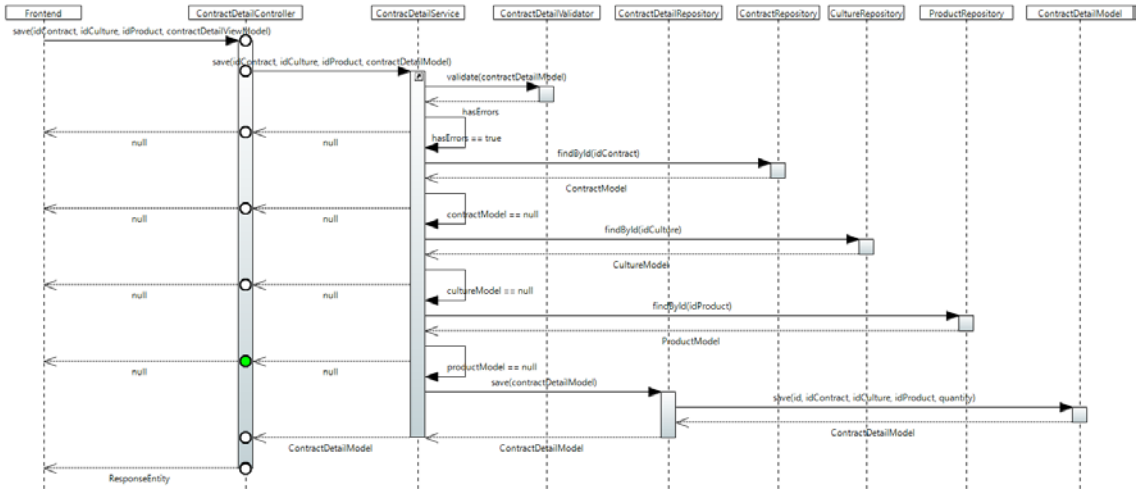


Ilustración 58: Diagrama de secuencia para agregar detalles a un contrato

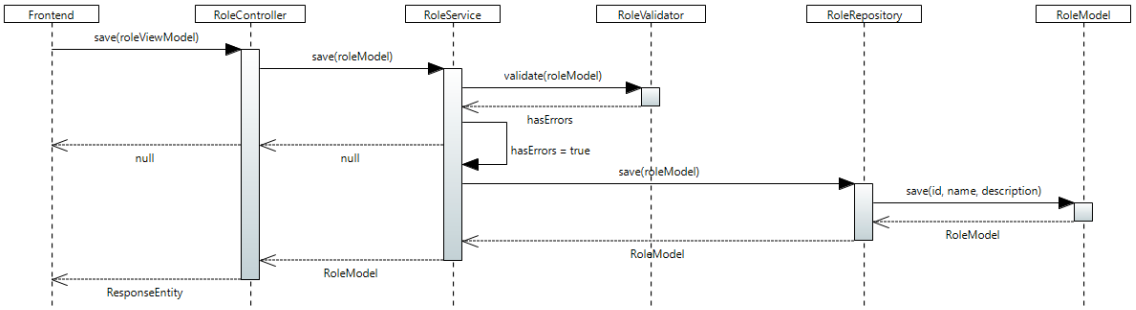


Ilustración 59: Diagrama de secuencia para agregar roles

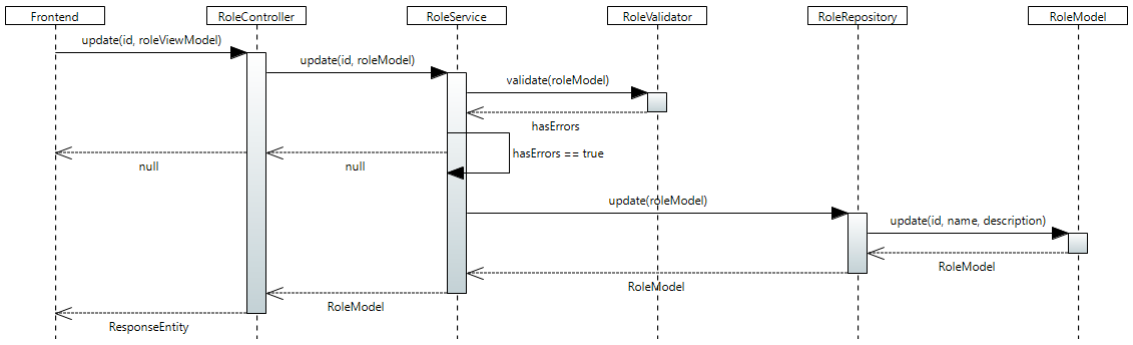


Ilustración 60: Diagrama de secuencia para editar roles

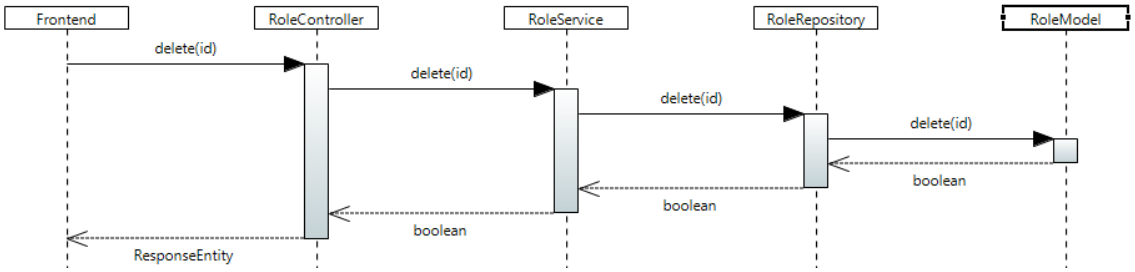


Ilustración 61: Diagrama de secuencia para eliminar roles

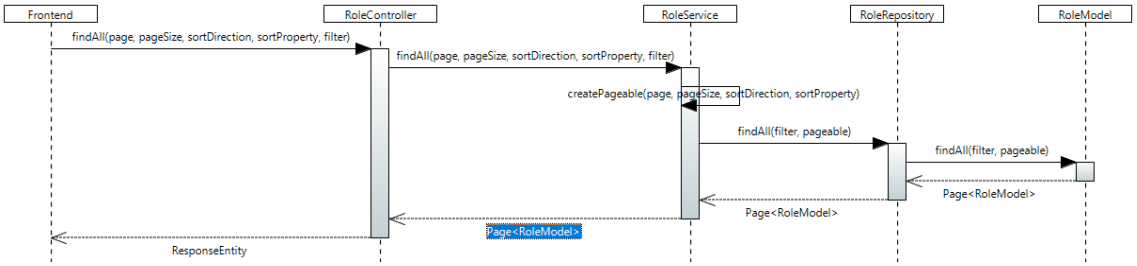


Ilustración 62: Diagrama de secuencia para listar roles

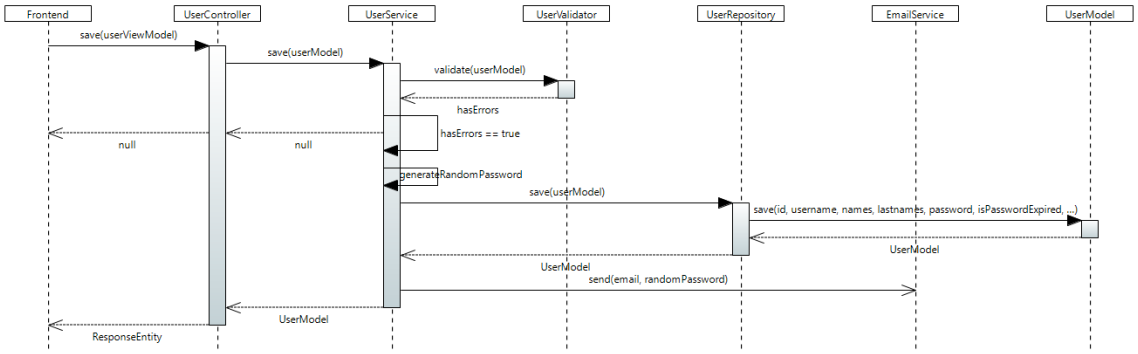


Ilustración 63: Diagrama de secuencia para agregar usuarios

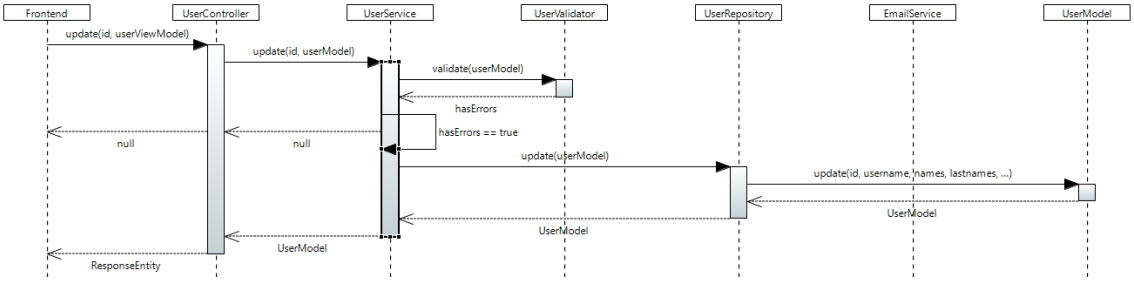


Ilustración 64: Diagrama de secuencia para editar usuarios

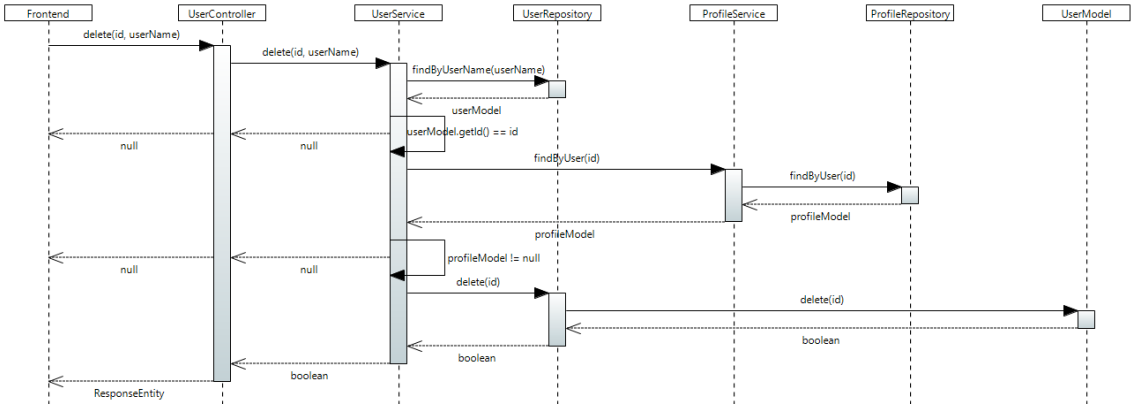


Ilustración 65: Diagrama de secuencia para eliminar usuarios

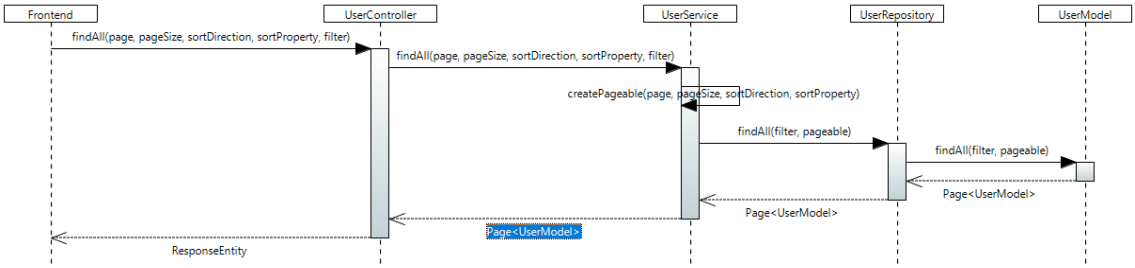


Ilustración 66: Diagrama de secuencia para listar usuarios

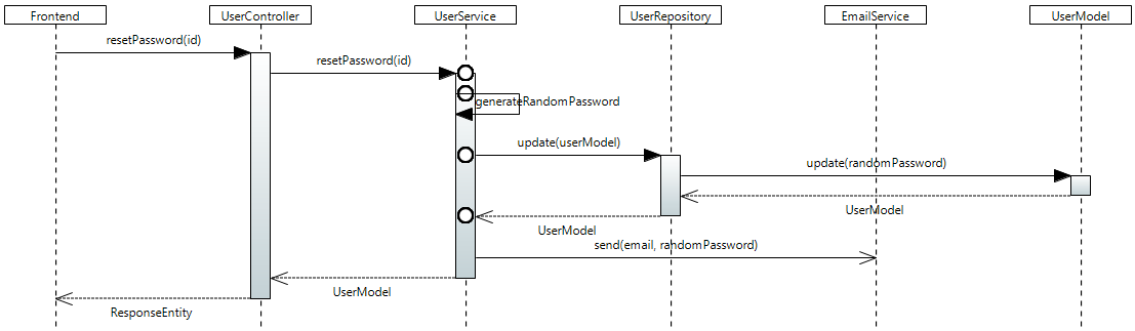


Ilustración 67: Diagrama de secuencia para resetear la contraseña de los usuarios

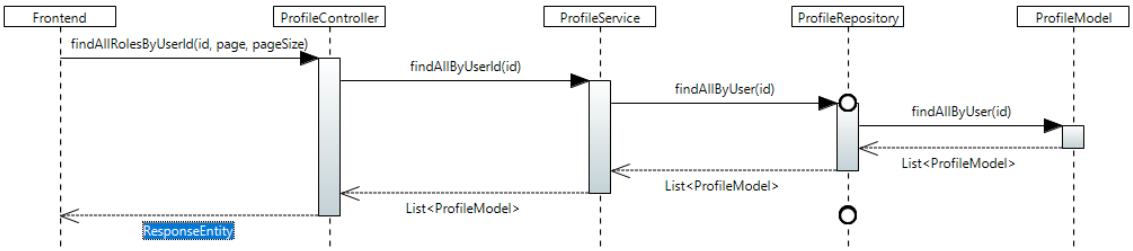


Ilustración 68: Diagrama de secuencia para listar los perfiles por usuario

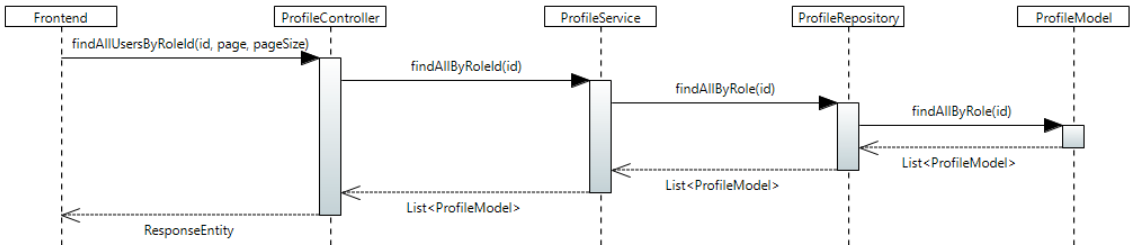


Ilustración 69: Diagrama de secuencia para listar los perfiles por rol

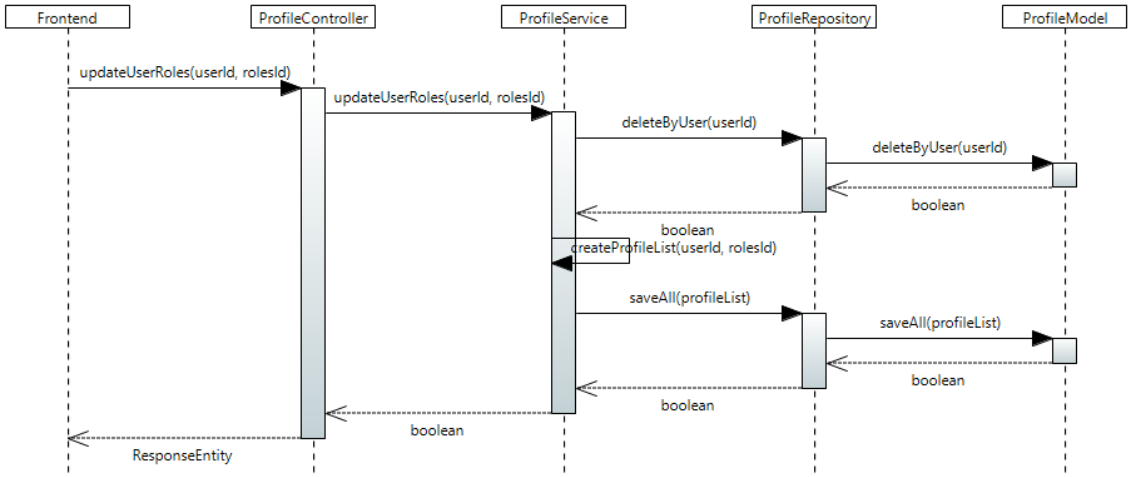


Ilustración 70: Diagrama de secuencia para actualizar los perfiles por usuario

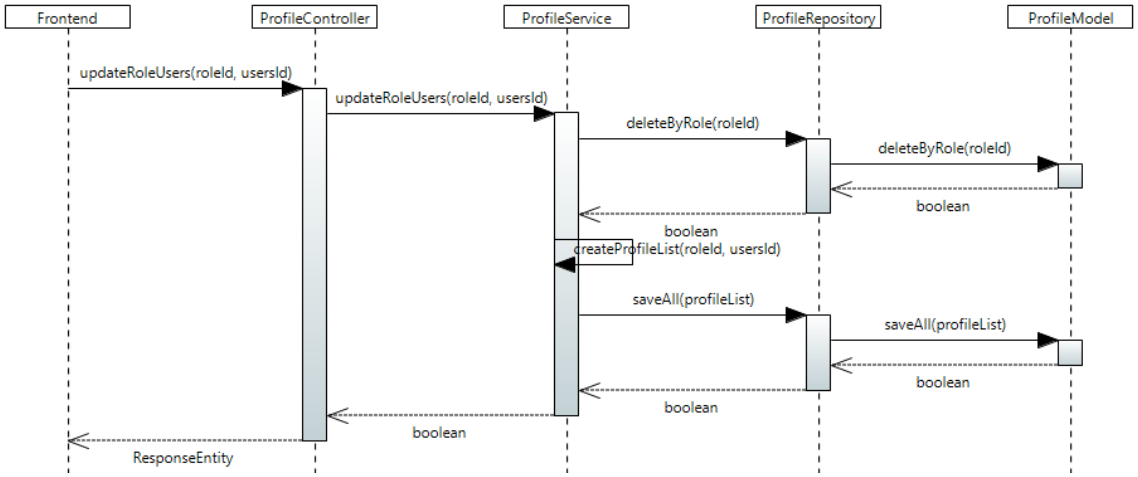


Ilustración 71: Diagrama de secuencia para actualizar los perfiles por rol

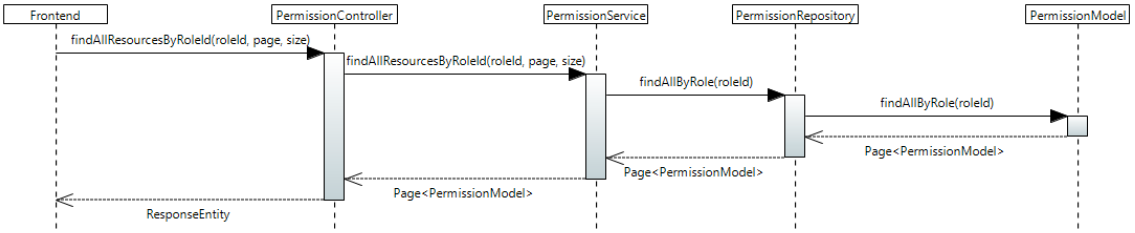


Ilustración 72: Diagrama de secuencia para listar los permisos por rol

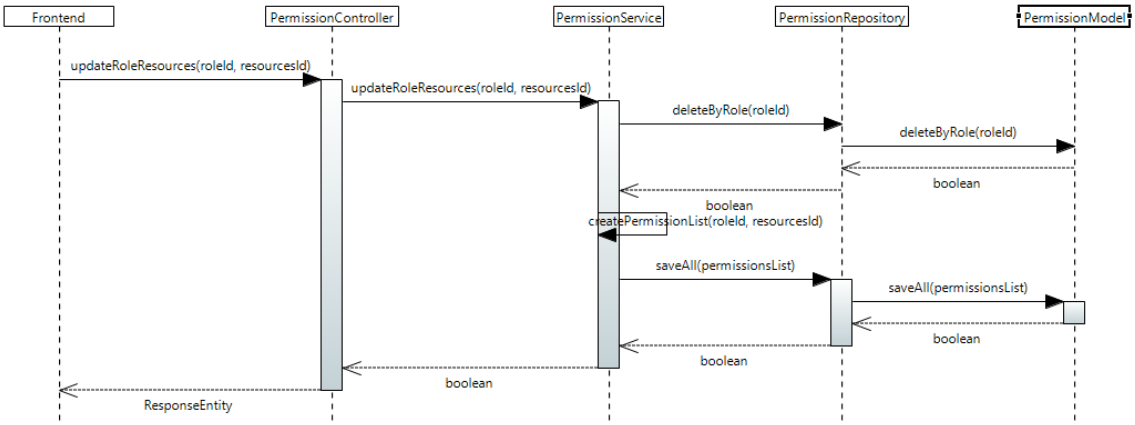


Ilustración 73: Diagrama de secuencia para actualizar los permisos por rol

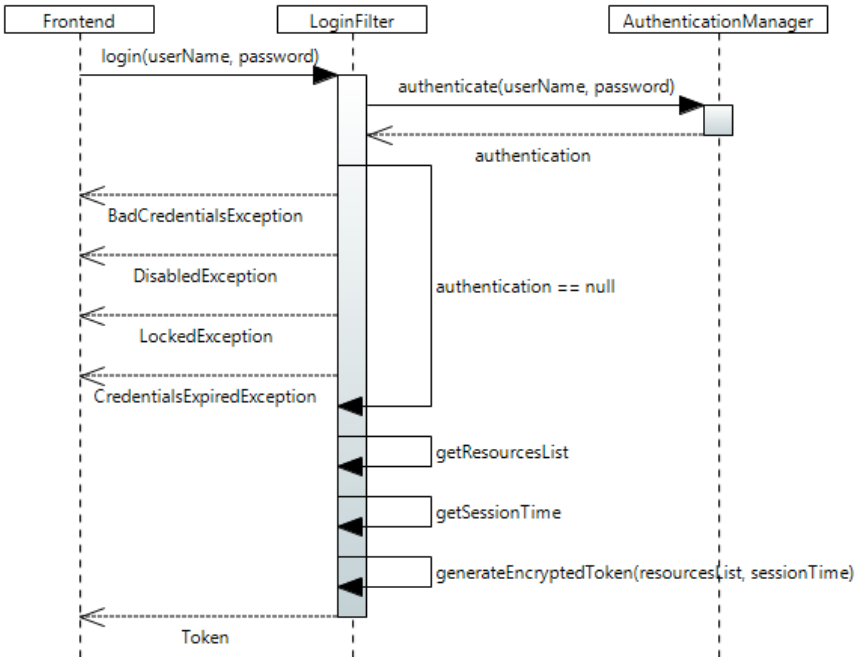


Ilustración 74: Diagrama de secuencia para la autenticación

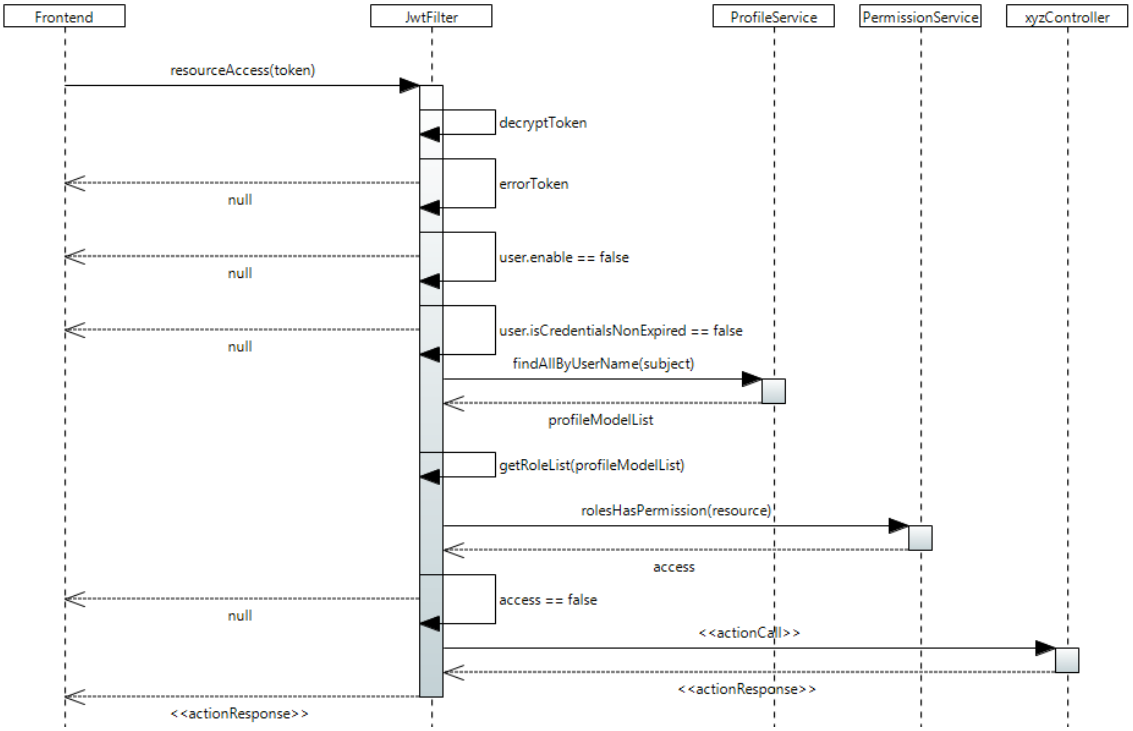


Ilustración 75: Diagrama de secuencia para el acceso a los recursos

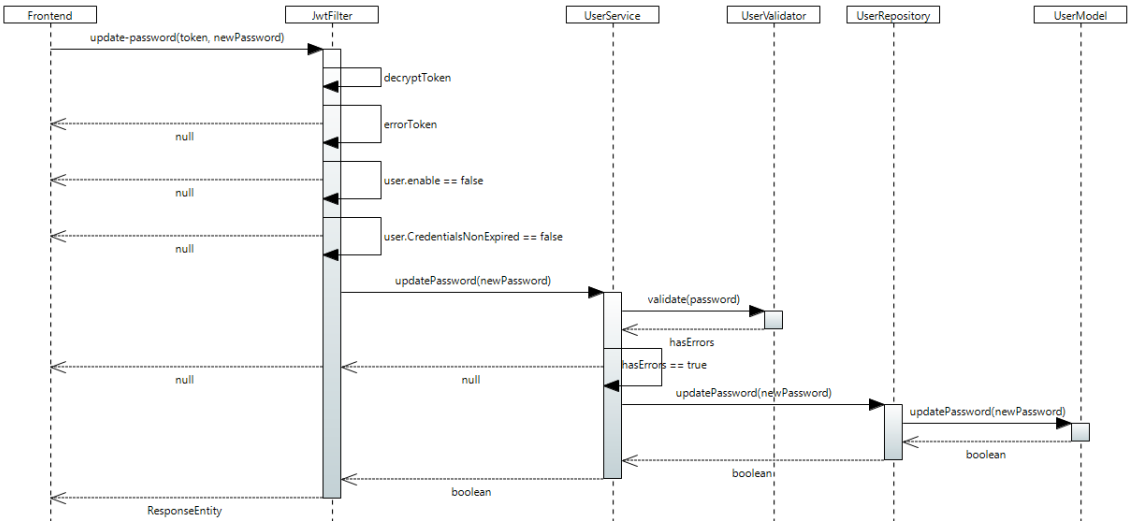
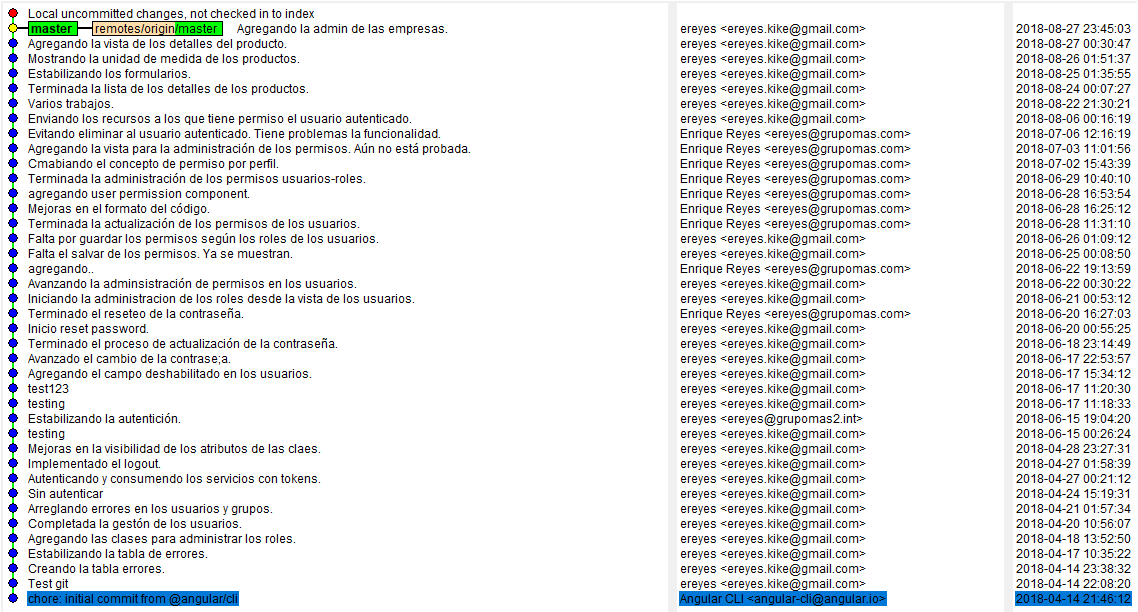


Ilustración 76: Diagrama de secuencia para actualizar la contraseña

Local uncommitted changes, not checked in to index		
master remotes/origin/master Agregando la cantidad de los productos y la cantidad de los cultivos a	ereyes <ereyes.kike@gmail.com>	2018-08-29 23:11:27
ACtualizando los contratos.	ereyes <ereyes.kike@gmail.com>	2018-08-29 20:33:23
Agregando la entidad contrato.	ereyes <ereyes.kike@gmail.com>	2018-08-29 19:34:10
Agregando la admin de las empresas.	ereyes <ereyes.kike@gmail.com>	2018-08-27 23:44:39
Agregando la vista de los detalles del producto.	ereyes <ereyes.kike@gmail.com>	2018-08-27 00:30:31
Mostrando la unidad de medida de los productos.	ereyes <ereyes.kike@gmail.com>	2018-08-26 01:51:14
Estabilizando los formularios.	ereyes <ereyes.kike@gmail.com>	2018-08-25 01:36:26
Terminada la lista de los detalles de los productos.	ereyes <ereyes.kike@gmail.com>	2018-08-24 00:07:08
Varios trabajos	ereyes <ereyes.kike@gmail.com>	2018-08-22 21:31:23
Enviando los recursos a los que tiene permiso el usuario autenticado.	ereyes <ereyes.kike@gmail.com>	2018-08-06 00:15:56
Actualizando el modelo de los permisos y de los perfiles.	ereyes <ereyes.kike@gmail.com>	2018-07-13 00:08:05
Evitando eliminar al usuario autenticado. Tiene problemas la	Enrique Reyes <ereyes@grupomas.com>	2018-07-06 12:16:47
Quitando la restricción de path único a los recursos.	Enrique Reyes <ereyes@grupomas.com>	2018-07-03 11:02:49
Agregando los permisos, existe error al agregar los recursos y los	Enrique Reyes <ereyes@grupomas.com>	2018-07-02 17:29:42
Cambiando el concepto de permiso por perfil.	Enrique Reyes <ereyes@grupomas.com>	2018-07-02 15:44:08
Terminada la administración de los permisos usuarios-roles.	Enrique Reyes <ereyes@grupomas.com>	2018-06-29 10:39:47
Terminada la actualización de los permisos de los usuarios.	Enrique Reyes <ereyes@grupomas.com>	2018-06-28 11:31:24
Agregado el mapeo para los roles y los errores.	Enrique Reyes <ereyes@grupomas.com>	2018-06-27 17:07:53
Agregando y configurando la clase UserViewModel que se utilizará como	Enrique Reyes <ereyes@grupomas.com>	2018-06-27 15:17:49
Mapeando los permisos. Se deben mapear los demas objetos enviados al	Enrique Reyes <ereyes@grupomas.com>	2018-06-25 19:00:11
Falta el salvar de los permisos. Ya se muestran.	ereyes <ereyes.kike@gmail.com>	2018-06-25 00:08:41
agregado...	Enrique Reyes <ereyes@grupomas.com>	2018-06-22 19:15:00
Avanzando la adminsistración de permisos en los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-22 00:30:36
Terminado sin probar la actualizacion de los permisos. Falta la	Enrique Reyes <ereyes@grupomas.com>	2018-06-21 19:08:39
Iniciando la administracion de los roles desde la vista de los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-21 00:53:51
Terminado el reseteo de la contraseña.	Enrique Reyes <ereyes@grupomas.com>	2018-06-20 16:26:18
Inicio reset password.	ereyes <ereyes.kike@gmail.com>	2018-06-20 00:55:42
Terminado el proceso de actualización de la contraseña.	ereyes <ereyes.kike@gmail.com>	2018-06-18 23:15:07
Adelantado proceso de actualizacion de la contrase;a.	Enrique Reyes <ereyes@grupomas.com>	2018-06-18 18:55:30
Avanzado el cambio de la contrase;a.	ereyes <ereyes.kike@gmail.com>	2018-06-17 22:53:48
Agregando el campo deshabilitado en los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-17 15:34:33
Estabilizando la autentición.	ereyes <ereyes@DESKTOP-VL3TP9T.grupomas2.int>	2018-06-15 19:02:59
Init 2.0	ereyes <ereyes@DESKTOP-VL3TP9T.grupomas2.int>	2018-06-12 18:11:26

Ilustración 77: Historial de los versionamientos en git del backend



Local uncommitted changes, not checked in to index		
master remotes/origin/master Agregando la admin de las empresas.	ereyes <ereyes.kike@gmail.com>	2018-08-27 23:45:03
Agregando la vista de los detalles del producto.	ereyes <ereyes.kike@gmail.com>	2018-08-27 00:30:47
Mostrando la unidad de medida de los productos.	ereyes <ereyes.kike@gmail.com>	2018-08-26 01:51:37
Estabilizando los formularios.	ereyes <ereyes.kike@gmail.com>	2018-08-25 01:35:55
Terminada la lista de los detalles de los productos.	ereyes <ereyes.kike@gmail.com>	2018-08-24 00:07:27
Varios trabajos.	ereyes <ereyes.kike@gmail.com>	2018-08-22 21:30:21
Enviando los recursos a los que tiene permiso el usuario autenticado.	ereyes <ereyes.kike@gmail.com>	2018-08-06 00:16:19
Evitando eliminar al usuario autenticado. Tiene problemas la funcionalidad.	Enrique Reyes <ereyes@grupomas.com>	2018-07-06 12:16:19
Agregando la vista para la administración de los permisos. Aún no está probada.	Enrique Reyes <ereyes@grupomas.com>	2018-07-03 11:01:56
Cmabiando el concepto de permiso por perfil.	Enrique Reyes <ereyes@grupomas.com>	2018-07-02 15:43:39
Terminada la administración de los permisos usuarios-roles.	Enrique Reyes <ereyes@grupomas.com>	2018-06-29 10:40:10
agregando user permission component.	Enrique Reyes <ereyes@grupomas.com>	2018-06-28 16:53:54
Mejoras en el formato del código.	Enrique Reyes <ereyes@grupomas.com>	2018-06-28 16:25:12
Terminada la actualización de los permisos de los usuarios.	Enrique Reyes <ereyes@grupomas.com>	2018-06-28 11:31:10
Falta por guardar los permisos según los roles de los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-26 01:09:12
Falta el salvar de los permisos. Ya se muestran.	ereyes <ereyes.kike@gmail.com>	2018-06-25 00:08:50
agregando..	Enrique Reyes <ereyes@grupomas.com>	2018-06-22 19:13:59
Avanzando la adminsistración de permisos en los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-22 00:30:22
Iniciando la administracion de los roles desde la vista de los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-21 00:53:12
Terminado el reseteo de la contraseña.	Enrique Reyes <ereyes@grupomas.com>	2018-06-20 16:27:03
Inicio reset password.	ereyes <ereyes.kike@gmail.com>	2018-06-20 00:55:25
Terminado el proceso de actualización de la contraseña.	ereyes <ereyes.kike@gmail.com>	2018-06-18 23:14:49
Avanzado el cambio de la contrase;a.	ereyes <ereyes.kike@gmail.com>	2018-06-17 22:53:57
Agregando el campo deshabilitado en los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-06-17 15:34:12
test123	ereyes <ereyes.kike@gmail.com>	2018-06-17 11:20:30
testing	ereyes <ereyes.kike@gmail.com>	2018-06-17 11:18:33
Estabilizando la autentición.	ereyes <ereyes@grupomas2.int>	2018-06-15 19:04:20
testing	ereyes <ereyes.kike@gmail.com>	2018-06-15 00:26:24
Mejoras en la visibilidad de los atributos de las claes.	ereyes <ereyes.kike@gmail.com>	2018-04-28 23:27:31
Implementado el logout.	ereyes <ereyes.kike@gmail.com>	2018-04-27 01:58:39
Autenticando y consumendo los servicios con tokens.	ereyes <ereyes.kike@gmail.com>	2018-04-27 00:21:12
Sin autenticar	ereyes <ereyes.kike@gmail.com>	2018-04-24 15:19:31
Arreglando errores en los usuarios y grupos.	ereyes <ereyes.kike@gmail.com>	2018-04-21 01:57:34
Completada la gestón de los usuarios.	ereyes <ereyes.kike@gmail.com>	2018-04-20 10:56:07
Agregando las clases para administrar los roles.	ereyes <ereyes.kike@gmail.com>	2018-04-18 13:52:50
Estabilizando la tabla de errores.	ereyes <ereyes.kike@gmail.com>	2018-04-17 10:35:22
Creando la tabla errores.	ereyes <ereyes.kike@gmail.com>	2018-04-14 23:38:32
Test git	ereyes <ereyes.kike@gmail.com>	2018-04-14 22:08:20
chore: Initial commit from @angular/cli	Angular CLI <angular-cli@angular.io>	2018-04-14 21:46:12

Ilustración 78: Historial de los versionamientos en git del frontend

Sistema para la gestión de proyectos de micropropagación de plantas biotecnológicas

Enrique Reyes Bermúdez
Área de Ingeniería y Tecnología
Universidad Internacional de la Rioja
Email: ereyes.kike@gmail.com

Abstract. *Micropropagation by tissue culture as a method of propagation is one of the most widely used techniques to increase the production of plants that are difficult to propagate through conventional methods. The bio-factories are used to obtain specimens through innovative scientific techniques that guarantee quality. The production of in vitro plants in commercial laboratories requires processes and operations of work closely related to each other, including the application of production planning methods that allow to define in advance the human and material requirements for its realization. Planning can be done through the use of appropriate software and can be organized using a network system that allows the linking of all areas. The following work exposes the design, development and results of a web application for the management of micropropagation projects of biotechnological plants, using essential elements and variables for production planning.*

I. INTRODUCCIÓN

El siguiente trabajo expone el diseño, desarrollo y resultados de una aplicación web para la gestión de proyectos de micropropagación de plantas biotecnológicas, utilizando elementos y variables indispensables para la planificación de la producción.

II. CONTEXTO

La biotecnología vegetal se ha desarrollado vertiginosamente desde las últimas décadas del pasado siglo, fundamentado por una parte en la obtención de plantas con caracteres mejorados a partir de la introgresión dirigida de genes deseables y, por otra, la introducción de nuevas las variedades mejoradas a la práctica agrícola, tolerantes o resistentes a factores bióticos o abióticos, lo cual requiere de protocolos eficientes de propagación de plantas a gran escala.

La aplicación de la micropropagación data del inicio de los años 70 del siglo XX, mayoritariamente en investigaciones con especies hortícolas. Sin embargo, cientos de tecnologías han sido desarrolladas para explotar el potencial genético de las plantas cultivadas.

La micropropagación de plantas se ha comercializado para producir plántulas de alta calidad y libres de virus. Alto costo y trabajos intensos son los principales problemas con esta técnica [1].

Los gastos directos de micropropagación se pueden dividir en dos grupos, que incluyen gastos de mano de obra, materiales y electricidad y la depreciación de la infraestructura. Los gastos directamente relacionados con la realización de una tecnología de propagación se definen como costos de operación. Estos podrían fácilmente cambiarse y están bajo el control constante del gerente de laboratorio [2].

Los costos de operación pueden dividirse en el trasplante de las plantas, la preparación de los medios y la limpieza de los recipientes de vidrio y de cultivo.

III. OBJETIVOS

OBJETIVO GENERAL

El siguiente trabajo tiene como objetivo general desarrollar una aplicación web para la gestión de proyectos de micropropagación a gran escala de plantas biotecnológicas y la generación de costos de producción bajo los principios del software libre.

OBJETIVO ESPECÍFICOS

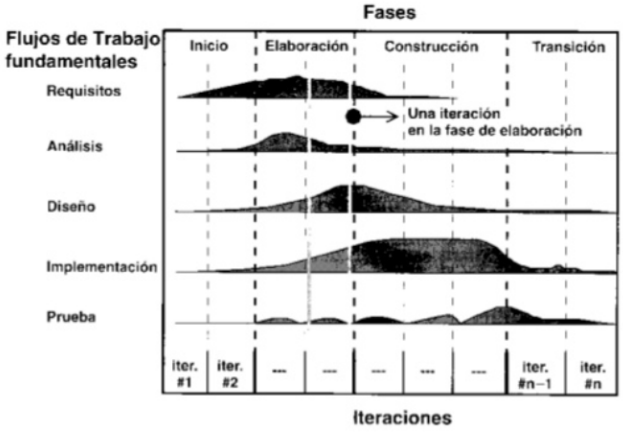
- Definir los requerimientos necesarios que debe cumplir el sistema para que sea totalmente operativo.
- Desarrollar la aplicación que responda a los requerimientos definidos y generar la documentación correspondiente que servirá como apoyo a las implementaciones y mejoras futuras.
- Realizar las pruebas a software desarrollado con el objetivo de validar sus funcionalidades.

IV. METODOLOGÍA DE TRABAJO

Un proceso de desarrollo de software se define como “el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software” [3, p. 4].

Según [4], el proceso unificado se apoya en UML para modelar todos los artefactos del software, definiendo los siguientes aspectos:

- ✓ Dirigido por casos de uso.
- ✓ Centrado en la arquitectura.
- ✓ Iterativo e incremental.



Fases y flujos de trabajo

V. REQUERIMIENTOS

[5, p. 108] define el concepto de requerimientos como “La descripción de los servicios proporcionados por el sistema y

sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información".

REQUERIMIENTOS FUNCIONALES

De acuerdo con [5, p. 109], los requerimientos funcionales "Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer".
El sistema desarrollado cumple con los siguientes requerimientos funcionales:

1. El sistema permitirá a los usuarios autenticarse mediante usuario y contraseña.
2. El sistema permitirá a los usuarios cerrar sesión.
3. El sistema permitirá al usuario actualizar su contraseña.
4. El sistema permitirá al administrador visualizar los logs.
5. El sistema permitirá al administrador gestionar los usuarios.
6. El sistema enviará al usuario la contraseña inicial por correo electrónico.
7. El sistema permitirá al administrador gestionar los roles.
8. El sistema permitirá al administrador gestionar los perfiles.
9. El sistema permitirá al administrador gestionar los permisos.
10. El sistema permitirá al administrador resetear la contraseña de los usuarios.
11. El sistema permitirá al administrador bloquear y desbloquear a los usuarios.
12. El sistema permitirá al usuario gestionar las empresas.
13. El sistema permitirá al usuario gestionar los cultivos.
14. El sistema permitirá al usuario gestionar los productos.
15. El sistema permitirá al usuario agregar detalles a los productos.
16. El sistema permitirá al usuario administrar los contratos.
17. El sistema verificará en cada requerimiento funcional si el token de sesión enviado por el usuario es válido, excepto para el RF1, RF2 y RF3.

REQUERIMIENTOS NO FUNCIONALES

Según [5, pp. 109,110], los requerimientos no funcionales "Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad."
El sistema desarrollado cumplirá con los siguientes requerimientos no funcionales:

1. Requerimiento de confiabilidad:
   a. Se debe garantizar que el sistema esté disponible las 24 horas del día.
   b. Las actividades de mantenimiento y supervisión del sistema no deben interferir en el correcto funcionamiento de la aplicación.
2. Requerimiento de seguridad:
   a. El sistema garantizará mantenerse en un estado seguro en caso de fallar algunas de sus funcionalidades.
   b. El sistema verificará si el usuario se encuentra autenticado antes de ejecutar alguna funcionalidad.
   c. El sistema verificará los permisos del usuario autenticado antes de ejecutar alguna funcionalidad.
   d. El sistema registrará las acciones solicitadas por los usuarios.
3. Requerimiento de portabilidad:
   a. El sistema debe permitir su despliegue tanto en la plataforma Windows como en Linux.
4. Requerimiento de apariencia o interfaz externa:
   a. El sistema debe validar todos los datos de entrada.
5. Restricciones de diseño e implementación:
   a. La implementación del sistema debe contar con un conjunto de patrones y principios de diseño que permita la reutilización de código, así como la facilidad de agregar nuevas funcionalidades.

VI. HERRAMIENTAS Y TECNOLOGÍAS

UML

"El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software" [6, p. 10].
Para obtener una mejor visión de la solución desarrollada se modelará el sistema a través de los siguientes diagramas:

- Casos de uso.
- Clases.
- Secuencia
- Paquetes.
- Despliegue.

JAVA

El lenguaje Java fue creado para ser utilizado en el mundo de los dispositivos electrónicos. En el año 1995 Java es utilizado como lenguaje de programación para computadoras, tras la fallida aceptación inicial de las empresas de electrodomésticos [7, p. 13].
Según [8, p. 11], la sintaxis del lenguaje Java es muy parecida a C y C++, aunque se evidencian las siguientes diferencias:

- ✓ No existen punteros, aumentando su seguridad y facilidad de uso.
- ✓ Es completamente orientado a objetos.
- ✓ Preparado para ser utilizado en aplicaciones de redes TCP/IP.
- ✓ Las excepciones son implementadas de forma nativa.
- ✓ Es un lenguaje interpretado, facilitando su ejecución remota y disminuyendo su rapidez en comparación con las aplicaciones escritas en lenguajes compilados como C++.
- ✓ Es multihilo, facilitando la ejecución de tareas en paralelo.
- ✓ Fuertemente tipado.
- ✓ Independiente de la plataforma, se ejecuta sobre una máquina virtual.

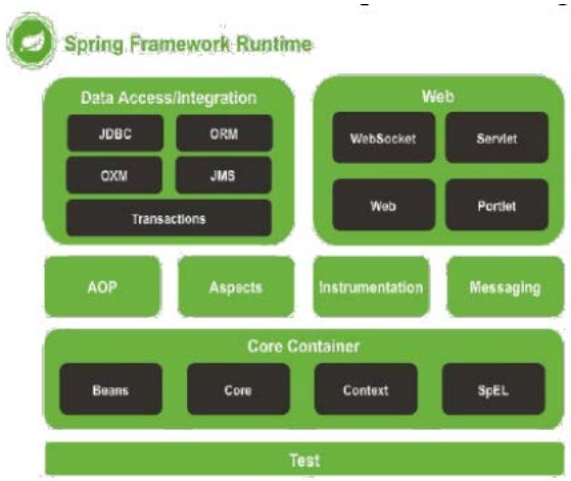
Q

Java es utilizado en multitud de ámbitos y tecnologías debido a sus características de seguridad, portabilidad, recurrencia, etc. En dependencia del ámbito donde se vaya a trabajar se pueden utilizar distintas plataformas: Java Card, Java Micro Edition, Java Standard Edition, Java Enterprise Edition. La plataforma Java EE implementa las APIs y las funcionalidades necesarias para poder ejecutar aplicaciones servidoras [9].

En el backend de la solución desarrollada se utilizará la plataforma Java EE para la construcción de un servicio que podrá ser consumido por cualquier cliente independientemente de la tecnología, lenguaje de programación, arquitectura, bibliotecas y framework seleccionados. La comunicación se podrá realizar siempre que se realicen correctamente las llamadas a las interfaces definidas por el servicio.

SPRING FRAMEWORK

Spring Framework es uno de los framework open source para el desarrollo de aplicaciones bajo la plataforma J2EE, permitiendo el desarrollo eficiente desde aplicaciones sencillas hasta las más complejas. Entre los principales conceptos que Spring Framework depende podemos encontrar la inyección de dependencia (DI), la programación orientada a aspectos (AOP) y la API de persistencia de Java (JPA). Spring framework está formado por casi 20 módulos. Entre sus módulos se encuentran Core Container, Data Access/Integration y Web. El módulo Web es el más importante para el desarrollo de aplicaciones web, integrado por los módulos Web, WebSockets, Portlet y Servlet. El módulo Servlet contiene dos de los conceptos más utilizados actualmente: Spring Model-View-Controller (MVC) y RESTful WebSerive (REST WS). Mediante Spring MVC se puede desarrollar aplicaciones en varias capas, permitiendo su expansión y denegando su modificación. REST WS permite la interoperabilidad entre sistemas informáticos, convirtiéndose hoy día en uno de los mecanismos más utilizadas para proveer datos a diversos tipos de consumidores [10].



Arquitectura de Spring Framework

ANGULAR

Angular es un framework JavaScript del lado cliente desarrollado por Google. La versión inicial de Angular se nombró AngularJS, luego se rescribió el framework completamente bajo el nombre de Angular 2. La versión actual de Angular es la 6, pero se refiere a la arquitectura de Angular 2, es decir, AngularJS es un producto distinto a Angular 2.

A través de una arquitectura MVC, Angular 2 facilita la implementación de aplicaciones web bien diseñada y bien estructuradas, proporcionando todas las funcionalidades que permitan manejar las entradas del usuario en el navegador, manipular los datos en el cliente y controlar cómo se muestran los elementos en el navegador.

Entre sus principales beneficios podemos mencionar:

- ✓ Facilidad de vincular datos a elementos HTML a través del “Data binding”.
- ✓ Extensibilidad, permitiéndole al usuario personalizar su propia implementación.
- ✓ Estricta obligación al usuario para escribir código limpio y lógico
- ✓ Flexibilidad para escribir código reutilizable.
- ✓ Soporte a través de Google como anfitrión.
- ✓ Compatibilidad al establecer una estrecha relación con el estándar de JavaScript [11].

En la solución propuesta se utiliza a Angular 2 como framework de desarrollo en el frontend, con el apoyo de los diferentes paquetes que facilitan la reutilización de código.

POSTGRESQL

PostgreSQL es un poderoso sistema de base de datos relacional open source que utiliza y extiende el lenguaje SQL. Cuenta con más de 30 años de desarrollo, remontándose en sus orígenes como parte del proyecto POSTGRES en el año 1986 dentro de la Universidad de California en Berkeley. PostgreSQL se puede utilizar en todos los sistemas operativos y tiene potentes complementos, como la popular extensión de base de datos geoespaciales PostGIS. PostgreSQL es altamente extensible, ofreciendo la posibilidad al desarrollador de definir sus propios tipos de datos, desarrollar funciones personalizadas y escribir códigos en diferentes lenguajes de programación sin la necesidad de recompilar su base de datos. Intenta cumplir además con el standard SQL y muchas de las funciones requeridas son compatibles, cumpliendo con al menos 160 de las 179 características obligatorias para SQL a partir del lanzamiento de la versión 10 en octubre del 2017, donde a partir de ese momento ninguna de las bases de datos relacional cumple en su totalidad con el estándar [12].

ECLIPSE

Eclipse es un IDE gratuito para desarrolladores escrito principalmente en Java. Es uno de los IDE más utilizados para Java, permitiendo el desarrollo de aplicaciones para varias plataformas (aplicaciones de escritorio, aplicaciones móviles, aplicaciones web y software empotrado). El proyecto Eclipse está operable dentro de los sistemas operativos Windows, Mac OSX y Linux, bajo una licencia pública de Eclipse y de código abierto (open source). Eclipse está apoyado por una comunidad de desarrolladores que facilitan su documentación y desarrollan plugins útiles para el desarrollo de software en todas sus etapas. Se puede considerar como el principal IDE en general ya que permite la programación en otros lenguajes diferentes a Java, por ejemplo, C++, PHP, Python, JavaScript, entre otros. Entre sus plugins principales se encuentra Windows Builder para el desarrollo de interfaces gráficas, permitiendo además a los programadores adaptar el IDE a sus necesidades a través de su sistema extensible de plugins.

Entre los IDE gratuitos o de pago basados en Eclipse podemos encontrar a MyEclipse y RAD de IBM [13, p. 5].

VISUAL STUDIO CODE

Visual Studio Code es un ligero, pero potente editor de código fuente que se puede ejecutar dentro de los sistemas operativos Windows, MacOS y Linux. Incluye un soporte integrado para Node.js, JavaScript, TypeScript y variedad de extensiones para otros lenguajes como PHP, C#, C++, Java, Go, Python, entre otros [14].

Visual Studio Code incluye soporte de autocompletado a través de IntelliSense, una rica comprensión semántica, refactorización de código y excelentes herramientas para tecnologías web como HTML, CSS, y JSON. Se integra además con administradores de paquetes y repositorios, incluye tareas comunes que facilitan el trabajo cotidiano de los desarrolladores y se acopla con el control de versiones distribuido de GIT, ofreciendo herramientas para el análisis de las diferencias en el código fuente [15, p. 4].

VII. DESARROLLO

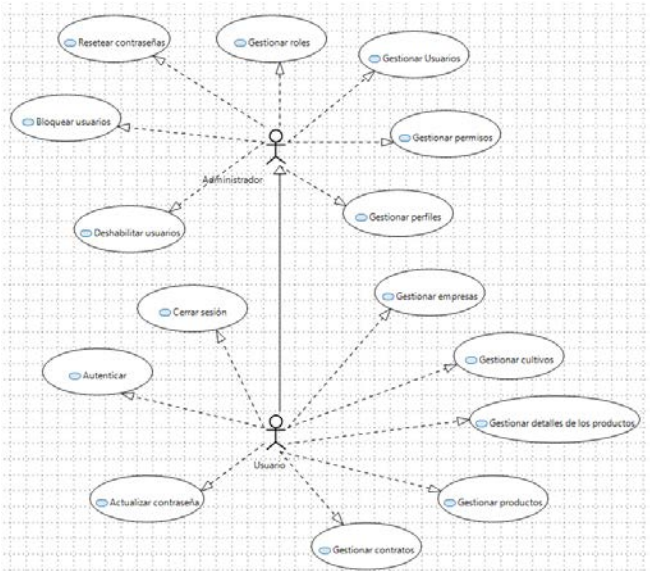
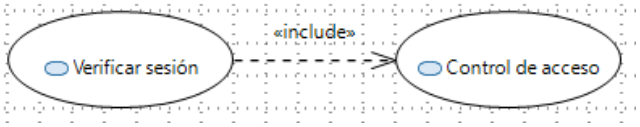


Diagrama de casos de uso

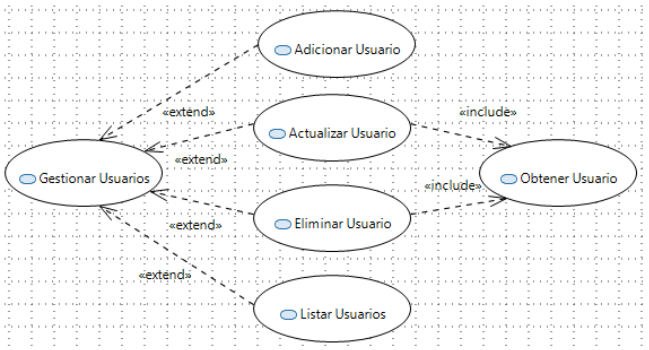
En la siguiente ilustración se detalla el caso de uso “Verificar sesión” y sus relaciones.



Caso de uso "Verificar sesión"

Excepto los casos de uso “Autenticar”, “Cerrar sesión” y “Actualizar contraseña”, todos los casos de uso tienen una relación de <inlcude> con el caso de uso “Verificar sesión”, que a su vez tiene una relación de <include> con el caso de uso “Control de acceso”. El objetivo del caso de uso “Verificar sesión” es validar que exista el token de seguridad enviado al cliente luego de su satisfactoria autenticación, donde el token debe ser enviado en la cabecera de cada petición realizada al servidor.

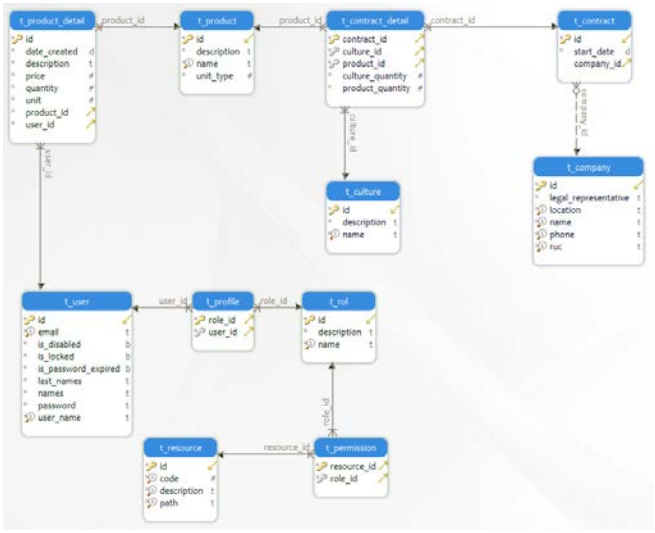
A continuación, se muestra un ejemplo de las relaciones que participan en cada uno de los casos de uso de tipo <Gestionar>, por ejemplo, “Gestionar Usuario”, “Gestionar Productos” y “Gestionar Contratos”.



Casos de uso relacionados a los CU tipo <Gestionar>

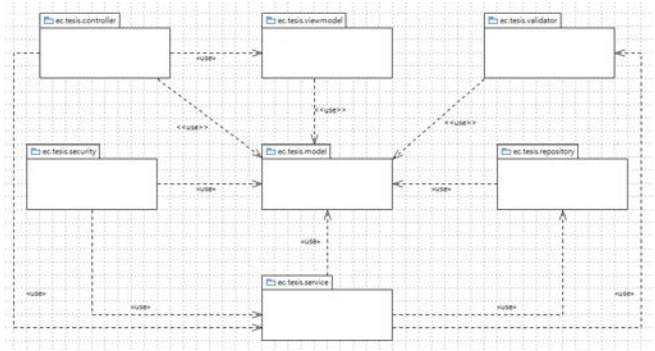
En el diagrama general no se muestran todas las relaciones para su mejor compresión. El usuario tiene la decisión de ejecutar las acciones de adicionar, actualizar, eliminar o listar sobre cada caso de uso de tipo <Gestionar>, estableciendo una relación de <extend> sobre el caso de uso original. A su vez existe una relación de <include> sobre la acción de obtener un elemento antes de ser actualizado o eliminado.

En la siguiente figura se muestra el modelo relacional de la base de datos utilizada en el aplicativo, detallándose las tablas y campos, sus relaciones, así como las llaves primarias y foráneas.



Modelo relacional

En la siguiente ilustración se muestran los diferentes paquetes de software que contiene el backend del aplicativo y sus diferentes relaciones.



VIII. EVALUACIÓN

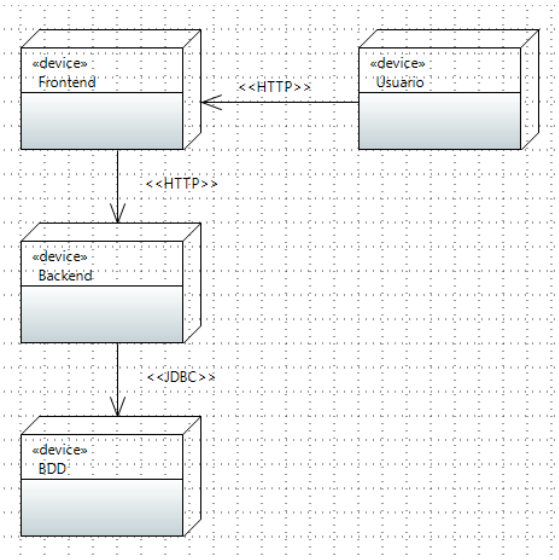


Diagrama de despliegue

SEGURIDAD

La seguridad es un tema inherente al desarrollo de aplicaciones web. Un gran porciento de los atacantes persigue este tipo de aplicaciones, debido a la gran cantidad de información que se puede manejar y la delicadeza de la misma. Si un sistema informático comprometiera sus datos, podría provocar una reducción sustancial de sus clientes. Entre las diferentes vulnerabilidades de seguridad se encuentran los intentos de autenticación y el control de acceso sin los permisos adecuados, los ataques de inyección de datos a través de SQL Injection y los ataques de inyección de contenido mediante Cross-Site-Scripting. (La importancia de la seguridad en las aplicaciones web., n.d.)

El aplicativo desarrollado cuenta con varios mecanismos que fortalecen su seguridad, entre los cuales se pueden mencionar:

- ✓ El acceso a datos se realiza a través de las interfaces que provee la clase JpaRepository, perteneciente al framework de persistencia Hibernate. En ningún momento se acceden a los datos mediante consultas no parametrizadas.
- ✓ Correcta configuración del mapeo de los controladores y acceso a sus diferentes métodos a través de las anotaciones @RequestMapping y sus derivados @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, entre otras.
- ✓ Validaciones de las entradas de datos mediante las clases de tipo <Validator> correspondiente a cada una de las entidades definidas.
- ✓ Uso del framework Spring Security para la configuración de la autenticación y el control de acceso.
- ✓ Generación de un token válido luego de una satisfactoria autenticación, el cual será verificado en cada una las solicitudes realizadas al sistema.
- ✓ Validación del acceso a los recursos solicitados.
- ✓ Generación de logs que almacenan las acciones que ejecuta el sistema y permiten ser auditados.

IX. CONCLUSIONES

A partir de la investigación realizada para la elaboración del producto presentado, utilizando a RUP como metodología idónea para lograr una mejor comprensión de los objetivos planteados inicialmente, se arriba a las siguientes conclusiones:

- ➢ Se definieron los requerimientos que garantizaron la operatividad del aplicativo desarrollado.
- ➢ Se desarrolló una aplicación web que permite la gestión de proyectos de micropropagación de plantas biotecnológicas teniendo en cuenta las premisas del software libre.
- ➢ Se validó el sistema a través de diferentes pruebas funcionales de caja negra.

X. RECOMENDACIONES

Luego de culminado el desarrollo de la aplicación se realizan las siguientes recomendaciones:

- ➢ Desarrollar una aplicación móvil que brinde las funcionalidades claves implementadas en el ambiente web.
- ➢ Agregar otros métodos de autenticación que puedan fortalecer el acceso al sistema permitiendo una doble autenticación.
- ➢ Integrar algoritmos de inteligencia artificial al sistema que permita la predicción de nuevas configuraciones que maximicen la productividad del cliente.
- ➢ Mantener el sistema actualizado con las nuevas tendencias de frameworks y bibliotecas de desarrollo.

XI. BIBLIOGRAFÍA

[1] P. Orellana, M. Suárez-Castellá, R. Triana, Z. Sarría, M. Pons, M. León, M. González y Z. Pérez, 2008.

[2] M. Hempel, M. Хемпел y M. Хемпел, «Some Economical Aspects of Commercial Micropropagation,» 1986.

[3] I. Jacobson, G. Booch y J. Rumbaugh, El Proceso Unificado de Desarrollo de Software, 2000.

[4] UNIR, «Metodologías, Desarrollo y Calidad de la Ingeniería de Software (ISW) - PER5 2017-2018 - Tema 2,» 2018.

[5] Sommerville, Ingeniería del software, Séptima ed., 2005.

[6] C. Larman, UML y Patrones, 2003.

[7] M. Torres Remon, Desarrollo de aplicaciones con JAVA, 2013.

[8] j. Sánchez Asenjo, Programación básica en Lenguaje Java, 2009.

[9] J. M. Ordax Cassá y P. A. Ocaña Díaz-Ufano, Programación web en Java.

[10] Ž. Jovanović, D. Jagodić, D. Vujičić y S. Ranđić, «JAVA SPRING BOOT REST WEB SERVICE INTEGRATION WITH JAVA ARTIFICIAL INTELLIGENCE WEKA FRAMEWORK,» 2017.

[11] B. Dayley, B. Dayley y C. Dayley, Learning Angular: A Hands-On Guide to Angular 2 and Angular 4, 2018.

[12] G. T. P. G. Development, «WHAT IS POSTGRESQL?,» 2018. [En línea]. Available: https://www.postgresql.org/about/.

[13] UNIR, «Plataformas de Desarrollo de Software (ISW) - PER5 2017-2018 - Tema 2,» 2018.

[14] Microsoft, «Documentation for Visual Studio Code,» 2018. [En línea]. Available: https://code.visualstudio.com/docs.

[15] T. Kahlert y K. Giza, Visual Studio Code Tips & Tricks Vol. 1, 2016.

U