ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: USO DE TECNOLOGIAS PARA EL DESARROLLO DE APICACIONES WEB 3.0

CASO PRÁCTICO: APLICACIÓN WEB DE ANÁLISIS DE COSTOS POR ENSAYO PARA AGROCALIDAD.

AUTOR:

AVILA MORALES LUIS FRANKLIN

DIRECTOR: MSC. CAMPAÑA MAURICIO

SANGOLQUÍ

2016

CERTIFICACIÓN

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIECIAS DE LA COMPUTACION
CARRERA DE ING DE SISTEMAS E INFORMATICA

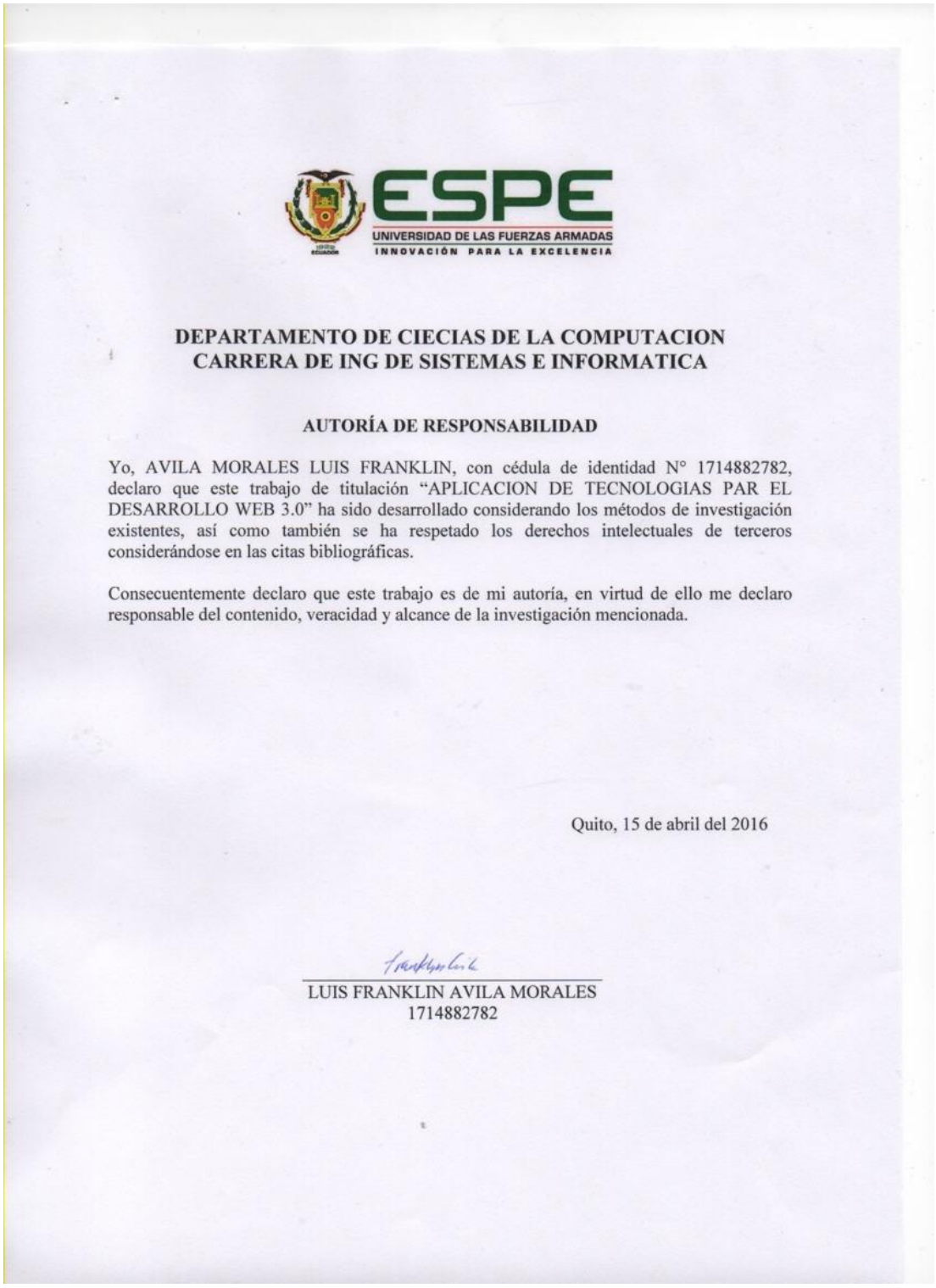
CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado en su totalidad por el Sr. LUIS FRANKLIN AVILA MORALES como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

Abril, 2016

DIRECTOR DE TESIS
ING. MAURICIO CAMPAÑA

AUTORÍA DE RESPONSABILIDAD



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIECIAS DE LA COMPUTACION
CARRERA DE ING DE SISTEMAS E INFORMATICA

AUTORÍA DE RESPONSABILIDAD

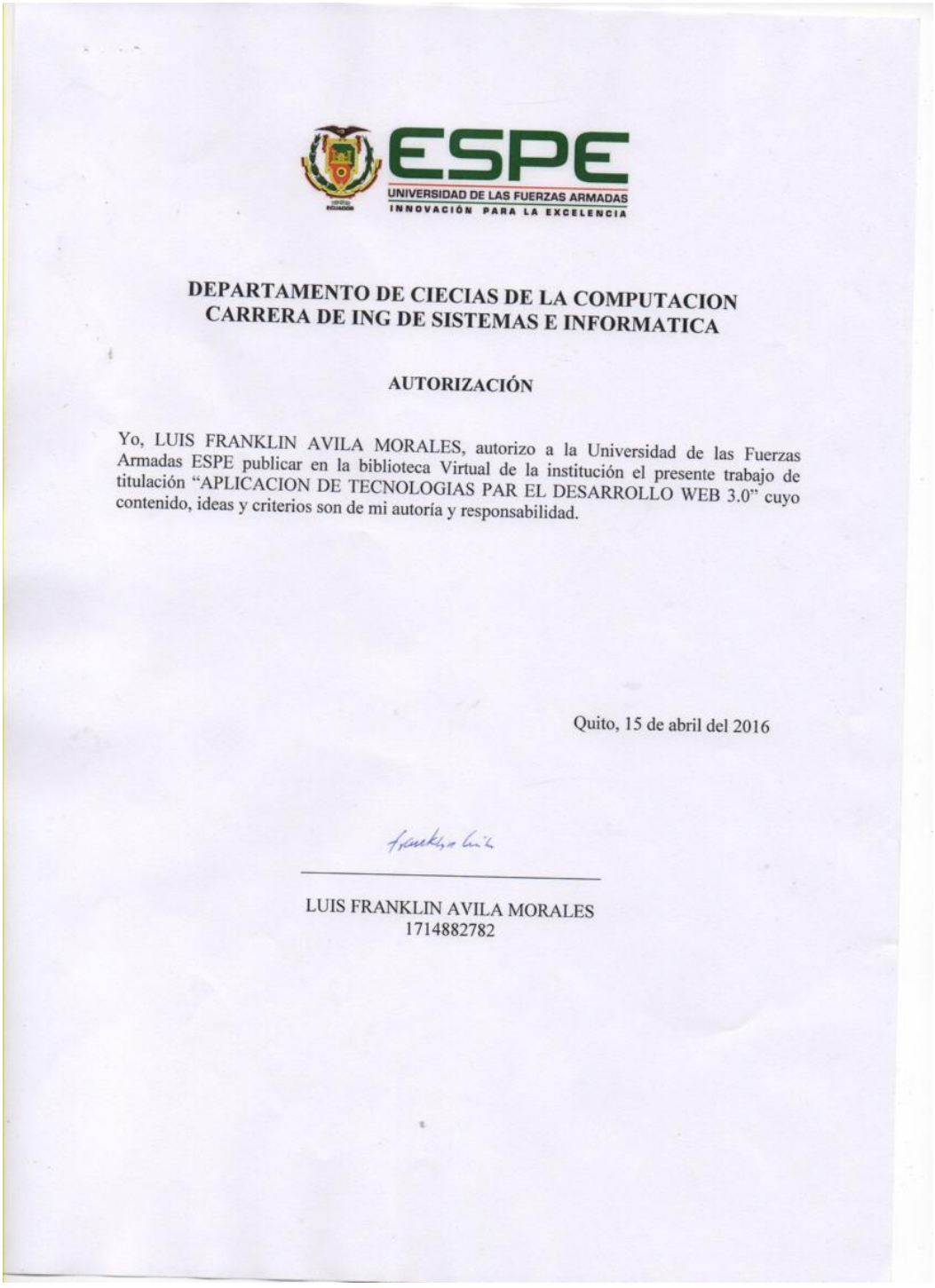
Yo, AVILA MORALES LUIS FRANKLIN, con cédula de identidad N° 1714882782, declaro que este trabajo de titulación "APLICACION DE TECNOLOGIAS PAR EL DESARROLLO WEB 3.0" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Quito, 15 de abril del 2016

LUIS FRANKLIN AVILA MORALES
1714882782

AUTORIZACION



DEPARTAMENTO DE CIECIAS DE LA COMPUTACION
CARRERA DE ING DE SISTEMAS E INFORMATICA

AUTORIZACIÓN

Yo, LUIS FRANKLIN AVILA MORALES, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "APLICACION DE TECNOLOGIAS PAR EL DESARROLLO WEB 3.0" cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Quito, 15 de abril del 2016

LUIS FRANKLIN AVILA MORALES
1714882782

DEDICATORIA

Este proyecto se lo dedico a mi Familia que son un ejemplo a seguir y siempre me han apoyado en todas las etapas de mi vida, siempre estaré agradecido por todo el apoyo incondicional y todo lo que he conseguido gracias a ellos.

Luis Franklin Ávila Morales

AGRADECIMIENTO

A mi familia por todo el apoyo y consejos que me han brindado, a la Universidad de las Fuerzas Armadas–ESPE por los conocimientos compartidos, a todos los Ingenieros y amigos que han aportado en mi crecimiento personal y académico.

Luis Franklin Ávila Morales

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La Tecnología web 3.0 tiene el propósito de dar una experiencia al usuario más rica y rápida en tiempo de respuesta, similar a una aplicación de escritorio. Agrocalidad presentó la necesidad de automatizar los procesos de cálculo de costos y presupuesto. Como consecuencia de no tener sistemas automatizados, el cálculo se lo realiza de forma manual, por lo cual estos procesos causan problemas en la planificación y toma de decisiones a gerencia. Para dar solución a este problema se creó AGROLAB, una aplicación web que gestiona la elaboración de presupuestos y estimación de costos, haciéndolos lo más cercano a la realidad. El sistema fue desarrollado utilizando el marco de trabajo Scrum, el framework de desarrollo AngularJS, framework de diseño Bootstrap y Postgresql como gestor de base de datos. El presente documento muestra los diferentes aspectos que convierten a AngularJS en un Framework muy efectivo en el desarrollo de aplicaciones web, recorriendo las capas que conforman la aplicación, las herramientas con las que cuenta el framework y mostrando cómo interactúan todos estos elementos. Como resultado se obtuvo un producto de software eficiente para la elaboración de presupuestos, con administración web y diseño adapativo. Se concluye que el uso de tecnologías web 3.0 mejora en la aplicación la transaccionabilidad, diseño, latencia y experiencia del usuario, con características de interoperabilidad y escalabilidad.

Palabras Clave:

- ANGULARJS
- AGROCALIDAD
- PRESUPUESTOS
- Web 3.0

ABSTRACT

Web 3.0 Technology has the purpose to provide a richer user experience and quick response time, similar to a desktop application. Agrocalidad presented the need to automate the process of costing and budget. As a result of not having automated processes, the calculation is done manually, so these processes cause problems in planning and decision-making management. To solve this problem, AGROLAB was created, a Web application that covers the process of budgeting and cost estimation, focused on the transparency of these processes, making them as close to reality. The system was developed using the Scrum framework, AngularJS, Bootstrap, and PostgreSQL as database manager. This document shows the different aspects that make a very effective AngularJS Framework in application development, covering the layers that make up the application, the tools available to the framework and showing how all these elements interact. As a result of efficient software product for budgeting, with administration web, and responsive design it was obtained. In conclusion, the use of web 3.0 technologies improves the transactionality, design, latency and user experience with interoperability and scalability features.

KeyWords:

- ANGULARJS
- AGROCALIDAD
- BUDGET
- Web 3.0

CAPITULO 1: PLANTEAMIENTO DEL PROBLEMA DE INVESTIGACION

1.1 Introducción

Los laboratorios de Agrocalidad se encuentran ubicados en el sector de Tumbaco, brindando servicios de análisis y diagnóstico en diferentes áreas del sector agropecuario desde el año 2008, donde cuentan con siete laboratorios especializados para realizar los diferentes análisis.

De la investigación realizada, en uno de sus procesos de suma importancia está el presupuesto por ensayo, el cual lo realiza cada laboratorio.

Actualmente el laboratorios de Agro calidad llevan el proceso de elaboración de presupuestos por ensayo manualmente, se dispone de archivos en Excel los cuales contienen información de materiales, insumos y reactivos que son utilizados para calcular el costo por cada ensayo, cada laboratorio es responsable de realizar su presupuesto, al final reúnen todos los archivos de Excel y lo unifican.

La problemática citada se evidencia en situaciones como al solicitar un reporte de proyección de gastos, sacar indicadores por cada ensayo realizado, realizar un informe de materiales utilizados o simplemente al requerir un historial de gastos, lo que causa incomodidad en las autoridades solicitantes, pues se ven afectados por las molestias que provoca el uso de un sistema anticuado e inadecuado de archivo y control, en especial en lo

El no haber implementado en la institución una alternativa de solución a este problema, conlleva a que la institución desvíe muchos recursos en la elaboración de presupuestos por ensayo y no tener una base de datos genera serios problemas al momento de entregar información.

Es por eso que el análisis, diseño e implementación de un sistema con tecnología web, que automatice el proceso de elaboración de presupuesto por ensayo funcionando en la intranet de la institución permitirá dotar a la institución una herramienta de gestión y planificación al centralizar toda la información pertinente, disminuyendo la carga de trabajo del personal administrativo, facilitando la entrega de información.

1.2 Planteamiento del Problema

La estimación de costos es un factor importante en la ejecución de presupuestos entregados a los laboratorios de Agrocalidad, motivo por el cual la estimación debe ser analizada automatizada y lo más cercano a la realidad posible, pero la mayoría de estimaciones entregadas por los laboratorios dedicadas son erróneos y obligan a reajustar el proceso manual, a continuación en la Figura se presentan algunas de las causales para esta problemática.

Figura 1 Proceso Actual

El proceso se realiza de la siguiente manera, como se describe en la figura1.

1.3 Justificación

El presente proyecto tiene como finalidad principal entregar a la institución pública, una aplicación web que se convierta en una herramienta de gestión que permita realizar el presupuesto del laboratorio en relación al proceso de costos por ensayo.

La importancia en el desarrollo del proyecto se basa en la necesidad de automatizar el proceso institucional y tener un respaldo de información para fomentar la trasparencia de sus procesos generales fortaleciendo la imagen corporativa.

Las actividades administrativas de la institución se verían notablemente mejoradas, al disminuir la carga de trabajo que implica este proceso, el personal administrativo y de laboratorios optimizarían su tiempo de trabajo en los accionares propios del proceso de elaboración de presupuestos.

La intranet es un medio que otorga muchas facilidades de comunicación,

reduciendo tiempos en el trabajo de las personas encargadas de realizar dicha actividad, la incorporación de nuevas tecnologías en el tratamiento de la información beneficia a Agrocalidad dotando de herramientas de gestión para sus actividades y con esto procesos transparentes en su administración.

1.4 Objetivos

1.4.1 Objetivo General

- Desarrollar una aplicación web para automatizar el proceso de elaboración de presupuesto por ensayos para el laboratorio de Agrocalidad usando metodologías agiles y tecnología web moderna.

1.4.2 Objetivos Específicos

- Realizar una investigación previa del framework AngularJS para su uso en el desarrollo del presente proyecto.

- Realizar el análisis, el diseño e implementación de la aplicación Web para el proceso de presupuesto de ensayos en base a los requerimientos que se necesiten.

- Aplicar una metodología ágil para el desarrollo de la aplicación

- Generar un sistema que brinde a gerencia una herramienta útil para toma de decisiones

1.5 Alcance

El alcance del proyecto viene dado por la creación de una Aplicación Web que automatizará y atenderá los siguientes procesos básicos:

- Usuarios
    - Administración de Usuarios para el ingreso al Sistema
    - Perfiles y permisos de Usuarios
- Catálogos

- Áreas
- Laboratorio
- Insumo
- Presentación
- Proveedores

- Unidades
- Procesos

- Ensayos
  - Composición de Ensayos.
  - Administración de Ensayos.

- Presupuesto
  - Administración de costos por ensayo.
  - Estimación de costos.
- Reportes
  - Reportes del presupuesto.

1.6 Metodología

Se utilizará para el desarrollo de la Aplicación WEB, una metodología Ágil, Scrum con Xp, constituirán la metodología ágil utilizada para el análisis, implementación y documentación de sistemas modernos.

Scrum no es un marco de trabajo con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Es un marco en el que la gente puede hacer frente a los problemas complejos, mientras que la entrega de manera productiva y creativamente productos de valor más alto posible.

“Scrum : Un marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente” (Scrum Agile, 2013).

Scrum es:

- Ligero
- Fácil de entender
- Extremadamente difícil de llegar a dominar

Scrum se basa en la teoría de control de procesos empírica o empirismo.

El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce, emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. (Scrum Agile, 2013)

Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación.

1.7 Herramientas

El siguiente tema describe las herramientas utilizadas en el proyecto.

1.7.1 Software

Para la realización del sistema se hará uso de los siguientes programas:

Tabla 1

Herramientas

Controlador	Aplicación
Compilador	Spring Tool Suit
Servidor Web	Apache Tomcat
Administración Base de Datos	Postgres
Framework de diseño	Bootstrap 3
Framework front end	Angular
Framework back end	Java

1.8 Factibilidad

La factibilidad permite identificar características para que el proyecto llegue a su culminación sin afectarse por factores externos como solo los técnicos, económicos, y operativos.

1.8.1 Técnica

Técnicamente el proyecto es viable, ya que dispongo del conocimiento técnico necesario para cumplir el desarrollo del sistema, la institución se compromete a entregar todo el apoyo necesario.

1.8.2 Económica

La factibilidad económica es totalmente viable ya que la institución se compromete a dar los equipos necesarios para el desarrollo del proyecto, en cuanto a licencias optamos por el software libre.

1.8.3 Operativa

Operativamente el proyecto es viable, ya que dispongo del conocimiento, la disponibilidad de una oficina de la institución y la total apertura a la información involucrada en el proceso de levantamiento de requerimientos garantiza el perfecto establecimiento de las necesidades

Tabla 2

Costos

Descripción	Valor	Meses	Subtotal
Desarrollador 1	$1500	4	$6000
Laptop 1	$1380		$1,380
Impresora	$90		$90
Eclipse	$0.0		$0.00
Jboss As	$0.0		$0.00
Postgresql	$0.0		$0.00
Movilización	$60	3	$180
Servicios Básicos	$30	4	$120
Internet	$45	4	$180
Papelería	$5	4	$20
Total			$7970

CAPÍTULO 2:

MARCO TEORICO

2.1 El Manifiesto Ágil

El Manifiesto Ágil Disciplinado es una extensión del original “Manifiesto para desarrollar Software Ágil”, en el que se reflejan las filosofías detrás del marco de trabajo Scrum.

En su estudio se enfatiza:

“Valores: Individuos e interacciones por sobre procesos y herramientas, Soluciones consumibles por sobre documentación comprensible, colaboración de stakeholders por sobre negociación por contrato, respuesta al cambio por sobre seguir el plan al pie de la letra.” (Manifiesto Agil, 2011)

Principios: “Satisfacer a los clientes mediante la entrega rápida continua de soluciones de alto valor” (disciplinedagiledelivery.org, 2014).

Se accede a cambios en los requisitos, incluso al final del ciclo de vida de la entrega de la solución. Los procesos ágiles aprovechan el cambio para dar ventaja competitiva al cliente. Se entregan soluciones consumibles con frecuencia, entre 2 semanas y 2 meses, prefiriendo acortar la escala de tiempo.

“Los clientes y los desarrolladores trabajan juntos cada día durante el proyecto.

Se construyen equipos con personas motivadas. Trabajan bajo un entorno y con el soporte que necesitan. Existe confianza en ellos para hacer el trabajo. El método más eficiente y efectivo para comunicar información al

equipo, y dentro de éste es la conversación cara a cara.” (Manifiesto Agil, 2011)

Los procesos ágiles promueven la entrega sostenible. Los clientes, desarrolladores y usuarios deberían poder mantener un ritmo constante indefinidamente, y la atención continua a la excelencia técnica y al buen diseño fundamenta la agilidad.

Simplicidad “el arte de maximizar la cantidad de trabajo todavía no realizado” (Manifiesto Agil, 2011), es esencial juntamente con las mejores arquitecturas, requerimientos y diseños emergen de equipos con capacidad auto-organizativa.

El equipo reflexiona como ser más efectivo a intervalos regulares, como consiguiente, afina y reajusta su funcionamiento.

Existe un apalancamiento para la evolución de la productividad de los bienes de la empresa, con un trabajo conjunto con el equipo de desarrollo y la gente responsable de los bienes.

El equipo visualiza el flujo de trabajo y lo mantiene equilibrado, logrando así un flujo bajo o poca presión debida a que el trabajo en proceso se mantiene al mínimo con tareas en paralelo.

El ecosistema organizacional debe evolucionar de tal manera que se pueda reflejar y resaltar los esfuerzos dentro de equipos ágiles, y aun así, ser lo suficientemente flexible para poder dar soporte a otros equipos no ágiles o híbridos.

La adaptabilidad al cambio proporcionara al equipo de estrategias y experiencia que guiaran el desarrollo de las fases del proyecto.

Tabla 3

Comparación metodologías

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos
Pocos Roles, más genéricos y flexibles	Más Roles, más específicos
El contrato debe ser bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Orientada a proyectos pequeños. Corta duración	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos
La arquitectura se va definiendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso: roles, actividades y artefactos
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

2.2 Introducción a Scrum

Las fases de la metodología tradicional ahora pasan a ser tareas, lo serían las fases de requisitos del sistema, requisitos del software, análisis, diseño, construcción, pruebas, y se ejecutarían de forma secuencial. Normalmente a

lo largo de pequeñas iteraciones durante todo el desarrollo denominados Sprint. (Scrum Org, 2009)

Se empieza a trabajar sin el detalle cerrado de lo que se va a producir, se debe partir de la visión general. Mediante el descubrimiento paulatino durante el desarrollo, y las circunstancias que se irán produciendo en el entorno, formaran el detalle de forma paralela al desarrollo.

Con Scrum conseguiremos la planificación y seguimiento del proyecto, mientras que con Xp se gestionará la parte de Desarrollo del software mediante las historias de usuario.

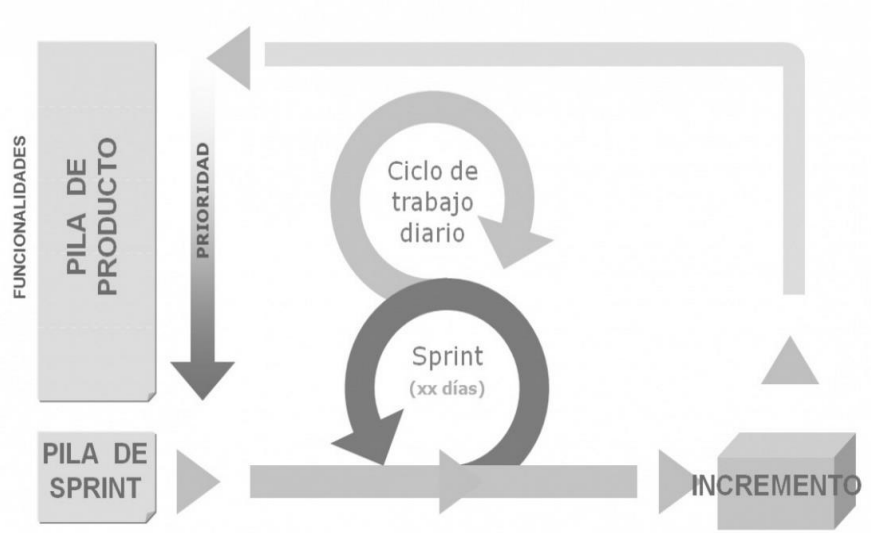


Figura 2 Ciclo Scrum

2.2.1 Definición de Scrum

“Scrum es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias

técnicas y procesos. Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo, de modo que podamos mejorar". (Scrum Org, 2009)

El objetivo de Scrum es elevar al máximo la productividad de un equipo mediante la auto-organización. Scrum se basa en lo que llaman un Sprint que es un esfuerzo concentrado para en un periodo de tiempo lograr metas fijadas. (Scrum Org, 2009)

Scrum son prácticas diseñadas por Schwaber y Sutherland para gestionar el desarrollo de software, las cuales están incluidas en la lista de modelos ágiles de Agile Alliance.

2.2.2 Roles

Scrum posee los roles de Scrum Team, product owner y srcum master.

El ScrumTeam denominado como equipo de trabajo, son los desarrolladores del proyecto de software, ellos deciden como será realizado el trabajo y como distribuir las asignaciones o requerimientos y cuánto van a tardar.

El ProductOwner representa la voz del cliente, trabaja con el ScrumTeam desde una perspectiva del negocio, administra un Product Back log que es una lista de tareas con especificaciones de un producto, prioriza las funcionalidades posibles.

El Scrum Máster mantiene procesos y trabaja como el director del proyecto en las reuniones diarias que mantiene, su función es eliminar los impedimentos posibles para el cumplimiento de objetivos de cada Sprint.

2.2.3 Artefactos

Los artefactos de Scrum son simples, representan trabajo o valor para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, que es necesaria para asegurar que todos tengan el mismo entendimiento del artefacto.

2.2.4 Lista de Producto (Product Backlog)

La lista de producto mantiene un orden de todo lo que podría ser necesario en el producto, es la única fuente de requisitos para cualquier cambio a realizarse en el producto, el responsable de la lista de producto es el dueño del producto.

Una lista de producto nunca está completa. El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio.

Para scrum mientras el desarrollo del producto exista, la lista de producto también existe. “La lista de producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a ser hechos sobre el producto para entregas futuras” (Garzas, 2015). Los elementos de la lista de producto tienen como atributos la descripción, el orden, la estimación y el valor, la lista de producto a menudo se convierte en una lista más larga, los requisitos nunca dejan de cambiar, así que la lista de producto es un artefacto vivo. Los cambios en los requisitos de negocio, las condiciones del mercado o la tecnología podrían causar cambios en la lista de producto.

Para describir el trabajo a realizar sobre el producto, se utiliza una única Lista de Producto. En ese caso podría emplearse un atributo de la lista de producto para agrupar los elementos. El refinamiento de la lista de producto consiste en: añadir detalle, estimaciones y orden a los elementos, es un proceso continuo, en el cual el Dueño de Producto y el Equipo de Desarrollo discuten acerca de los detalles cada uno de los elementos de la Lista de Producto.

2.2.5 Lista de Pendientes del Sprint (Sprint Backlog)

La Lista de Pendientes del Sprint son las denominadas tareas de la Lista de Producto seleccionados para un Sprint. La Lista de Pendientes del Sprint es una predicción realizada únicamente por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo que implicara ejecutarlo. En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

ProductOwner recoge todas las peticiones y especificaciones que son la base de los cambios del producto, pueden ser correcciones de errores y nuevas funcionalidades

Al mismo tiempo se hace una lista de prioridades, el Product owner toma las decisiones personalmente acerca de prioridades en los cambios y entregas.

2.2.6 Eventos de Scrum

2.2.7 Fase Sprint

El tiempo de cada Sprint es de 2 a 4 semanas, en esta fase se crea primeramente un Backlog, y cuando las tareas y el tiempo requerido son definidos el product owner pone en marcha la fase.

En esta fase el ScrumTeam trabaja por su propia responsabilidad, al combinarlo con Extreme programming se adaptara su prácticas de ingeniería para el análisis, diseño, desarrollo y la entrega del producto ayudándonos con diagramas de UML.

2.2.8 Scrum Diario

Todos los días el ScrumTeam y el Scrum Máster deben tener una breve reunión tratando tres preguntas esenciales

- ¿Qué has hecho desde la última reunión?
- ¿Qué harás hoy hasta la próxima reunión?
- ¿Hay algo que impida lo que haces según lo tenías planeado?

Las dos primeras preguntas dan una visión a los participantes de cómo están avanzando en el proyecto, la tercera pregunta sirve de base pilas para solucionar el problema, todos los interesados en el proyecto pueden estar en la reunión pero solo el ScrumTeam y el Scrum Máster pueden hablar.

2.2.9 Demostración

Cada Sprint termina con una demostración en un gráfico burn-down que marca el día a día y el trabajo programado en horas y objetivos cumplidos.

2.2.10 Retrospectiva

Se analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.

2.3 Extreme Programming

Extreme Programming es una disciplina de desarrollo de software basado en los valores de la simplicidad, la comunicación, la retroalimentación y coraje. Su acción consiste en llevar todo el equipo junto con la presencia de prácticas sencillas.1

2.3.1 Fases

- Exploración
  En esta fase, los clientes plantean a grandes rasgos las historias de usuario.
- Planificación de la Entrega (Release)
  En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas.
- Iteraciones
  Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado.

---

[1] (extremeprogramming.org, 2013)http://www.extremeprogramming.org/

- Producción

  Se realizan de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea puesto a producción.

Estas son recomendaciones que se deben seguir para lograr el trabajo acorde a la metodología ágil.

2.3.2 SCRUM+XP: prácticas

Extreme Programming mejora un proyecto de software en cinco formas esenciales; la comunicación, la sencillez, la retroalimentación, el respeto y el valor, implementa un entorno sencillo, pero eficaz permite a los equipos de convertirse en altamente productiva. El equipo se auto-organiza alrededor del problema para resolverlo lo más eficientemente posible.

- Planificación: ocurre durante la planificación del sprint: el product owner propone funcionalidades a desarrollar, y el equipo de desarrollo las estima en horas de trabajo.[2]
- Entregas pequeñas y frecuentes: duran 1 a 4 semanas.
- Historias de usuario: funcionalidades que apuntamos en el product backlog.
- Diseño simple: Programar el mínimo código posible que implementa una funcionalidad, sin adornos.
- Pruebas unitarias: Pruebas antes que el propio código, al menos cuando es posible.
- Refactorizaciones: Código en continua evolución.
- Integración continua: hacemos versiones frecuentes y automáticas.

[2] http://www.proyectalis.com/wp-content/scrum-y-xp-desde-las-trincheras.pdf

2.4 AngularJs

Hubo un tiempo, hace muchos años, cuando Internet cualquier tipo de lógica dentro de una página web tenía que ser enviado al servidor para su procesamiento y luego renderizado como una nueva página web. Este arreglo " llamar y actualizar" para una experiencia de usuario deficiente, que sólo se agrava cuando la latencia de red era especialmente alta.

Todo el paradigma se volcó con la introducción de XMLHttpRequest y la capacidad de hacer llamadas asíncronas al servidor sin tener que recargar la página. Esto hizo que para una experiencia de usuario mucho más coherente, porque el usuario podría realizar una tarea que requiere una llamada remota y aún interactuar con el solicitud que la llamada se está realizando y procesada. Aquí es donde JavaScript logro demostrar que el trabajo se podría hacer de una manera transparente.

Como resultado surgio AngularJS que es un framework JavaScript de código abierto, desarrollado por Google. Como se puede observar mediante estadísticas el uso del framework de AngularJs con el tiempo se va incrementando, además de tener una comunidad organizada de desarrolladores probando las funcionalidades del framework, reportando bugs.

Se puede observar las búsquedas de AngularJs con respecto a otros frameworks. En la figura 3 se encuentran descritos los framework javascript más utilizados en el mundo, desde el año 2011, posicionándose como el mejor estos dos últimos años.

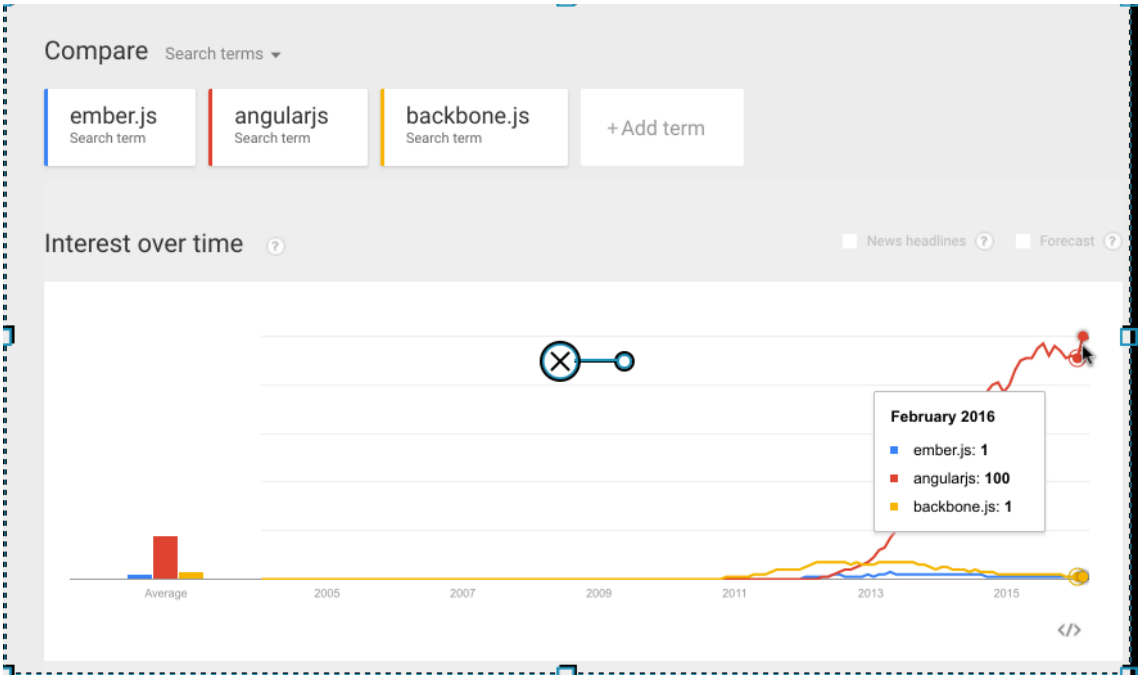


Figura 3 Comparación Angular JS

AngularJs proporciona un framework basado en el patrón MVC de JavaScript para el desarrollo de aplicaciones web en el lado del cliente Front End con la característica de SPA (Single-Page Applications).

Angular le da una gran flexibilidad a la lógica de presentación bien separada de la lógica de negocio y el estado de presentación.

Una de las características que decide su uso es la documentación del framework, ya que dispone de un sitio donde describen las bondades del framework.

2.4.1 Scopes

Los scopes son los distintos contextos de ejecución sobre los que trabajan las expresiones de AngularJS, por ejemplo, cuando referenciamos un atributo del modelo mediante la directive ng-model, no estamos sino

apuntando a un atributo que contiene el scope sobre el que se está trabajando. En los scopes se guarda la información de los modelos que se representan en la vista y también atributos que se utilizan para manejar la lógica de la misma.

Los scopes se manejan principalmente desde los controladores o desde las directivas.

2.4.2 Controllers

Los controladores son los delegados para inicializar y modificar la información que brindan los scopes, en función de las necesidades de la aplicación, también podemos declarar funciones en el scope que se podrán utilizar más tarde o ser llamadas desde la vista.

2.4.3 Client-side template

En un sistema de plantillas por lo general es el servidor el encargado de mezclar la plantilla con los datos y devolver el resultado al navegador. En la filosofía de AngularJS “el servidor proporciona los contenidos estáticos (plantillas) y la información que se va a representar (modelo) y es el cliente el encargado de mezclar la información del modelo con la plantilla para generar la vista” (www.educacionit.ar, 2015).

2.4.4 Data binding

Con AngularJS podemos sincronizar el modelo y la vista automáticamente utilizando ciertas directives del framework. Esta sincronización es bidireccional, es decir, la información se sincroniza tanto si cambia el valor en la vista como si lo hace el valor en el modelo.

2.4.5 Directivas

Una directiva es un marcador de etiquetas HTML, le dicen a angular que estas directivas se van a ejecutar o son referencias de algún código en javascript, mediante el uso de directivas del framework o personalizadas se extiende la sintaxis de HTML y manejar el comportamiento que se necesita. Podemos crear directivas a nivel de elemento, de atributo, de clase y de comentario.

2.4.6 Filtros

Los filtros modifican la forma en la que presenta la información al usuario.

2.4.7 Servicios

Los servicios son los encargados de comunicarse con el servidor para enviar y obtener información que después será tratada por los controladores para mostrar datos en vista.

Esta parte es más compleja de explicar con un ejemplo, por el momento nos basta con saber que los servicios se pueden dividir en tres categorias: servicios, factorias y proveedores.

El servicio $resource incluido en el framework permite encapsular la interacción con servicios RESTFULL evitando tratar directamente con las llamadas http.

Otros servicios interesantes que incluye AngularJS son $q y las llamadas promesas. Mediante este mecanismo podemos realizar acciones asíncronas y devolver valores que puede que aún no hayan sido resueltos. Cuando la acción ha finalizado el valor devuelto, llamado promesa, se resuelve en función del resultado de la misma, mientras tanto la ejecución del programa sigue su curso.

2.4.8 Módulos

Son piezas que forman parte de nuestra aplicación en angularJS, hacen el código mantenible, testeable y entendible, sirven para definir las dependencias entre módulos.

2.4.9 Inyección de dependencias

“La inyección de dependencias (en inglés Dependency Injection, DI) es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. El término fue acuñado por primera vez por Martin Fowler” (GARCIA, 2015).

La inyección de dependencias en AngularJS le permite describir de forma declarativa las dependencias entre los elementos que conforman una aplicación. O sea, que la aplicación no necesita ningún método main o punto de entrada. La inyección de dependencia es también un núcleo de

AngularJS. Esto significa que cualquier componente que no se ajuste a sus necesidades puede ser fácilmente reemplazado.

“Todos los componentes registrados en angular son singleton, es decir, sólo existe una instancia de ellos en la aplicación y si hay varios componentes que dependen de un mismo objeto, todos recibirán la misma instancia del objeto. Esto es lo que permite utilizar fácilmente los servicios para almacenar estado, ya que dos Controllers que dependan de un mismo servicio estarán utilizando el mismo objeto y, por tanto, podrán compartir información a través de él” (Hernández, 2013).

2.4.10 Ventajas

AngularJs es un framework intuitivo que facilita organizar el código, hay una urgente necesidad de ser capaz de organizar el código de una manera que ayuda al mantenimiento, la colaboración, la legibilidad y la extensión. AngularJS fue escrito desde el principio para ser testeable, y es probable que esta característica, junto con las decisiones de diseño que vinieron de este compromiso, haya jugado un papel muy importante en la adopción de AngularJS. Un framework que es propicio para escribir test.

Data Binding bidireccional ahorra cientos de líneas de código ya que con las dos vías de enlace de datos es la supermodelo del conjunto de características. Cuando se escribe aplicaciones con jQuery, se habría tenido que usar jQuery para consultar el DOM para encontrar un elemento específico, detectar un evento, analizar el valor del elemento DOM, y a continuación, realizar alguna acción en ese valor.

En AngularJS, simplemente tiene que definir una propiedad de JavaScript y luego se une a él en nuestro HTML, y ya está. Obviamente, hay algunas

variaciones a este escenario, pero no es raro escuchar de aplicaciones jQuery siendo reescrito y muchas líneas de JavaScript acaban de desaparecer. Al eliminar todo el código repetitivo que anteriormente se requería para mantener nuestra HTML y JavaScript en sincronía, se tiene la capacidad de realizar más trabajo en menos tiempo con significativamente menos esfuerzo.

Plantillas HTML, HTML es un lenguaje inherentemente limitada que fue diseñado para facilitar el diseño y la estructura, interacciones no complejos. En otras palabras, no fue creado para vivir en el mundo de la aplicación web moderna tal como la conocemos ahora. Algunos tratan de superar los marcos esta limitación mediante la abstracción a cabo enteramente en HTML en cadenas o algún preprocesador. AngularJS abraza HTML mientras que da a los desarrolladores la capacidad para superar sus limitaciones, extendiéndolo a hacer lo que es lo que necesitamos.

Estructuras de datos que son solo JAVASCRIPT permiten hacer de la integración un proceso más sencillo. Por otro lado, la posibilidad de trabajar con Plain Old objetos JavaScript (POJO) hace la integración con otras tecnologías muy sencillas. Al consumir y proporcionar JavaScript sin necesidad de envolver y desenvolver en con mecanismos de propiedad, AngularJS es capaz de consumir datos de otras fuentes mucho más eficientemente. Se puede hacer que los modelos enviados con JSON desde el servidor y sean consumidos al instante en la aplicación de AngularJS.

Hay algunas características muy interesantes de AngularJS que son bastante útiles, se trata de esbozar algunos puntos importantes de cómo AngularJS es un framework javascript en un sentido muy práctico.

Tabla 4

Estructura Angular

Componente	Propósito
Módulo	Sirven como un contenedor para ayudar a organizar el código dentro de la aplicación AngularJS.
Configuración	El bloque de configuración de una aplicación AngularJS permite la configuración para aplicar antes que la aplicación funcione realmente. Esto es útil para la creación de rutas, de forma dinámica la configuración de servicios.
Rutas	Las rutas permiten definir formas de navegar a estados específicos dentro de la aplicación. Ellos También le permiten definir opciones de configuración para cada ruta específica, como por ejemplo qué plantilla y un controlador para su uso.
Vista	La vista en AngularJS es lo que existe después de que AngularJS ha compilado y renderizado el DOM con todo JavaScript en su lugar.
Ámbito	Es el pegamento entre la vista y el controlador dentro de una aplicación AngularJS. Con la introducción del controlador como sintaxis, la necesidad de utilizar explícitamente $scope se ha reducido considerablemente.
Controlador	El controlador es responsable de definir los métodos y propiedades que se puede unir e interactuar con la vista, los controladores deben ser ligeros y sólo centrarse en la idea de que están controlando.
Directiva	Es la extensión de una vista en AngularJS ya que le permite crear elementos reutilizables que encapsulan cierto comportamiento. Se puede pensar en las directivas que sean componentes para el código HTML.
Servicio	Proporciona una funcionalidad común a una aplicación AngularJS. Los servicios se extienden a los controladores y los hacen más accesible a nivel de aplicación.

2.5 Boostrap

Es un framework que utiliza css, html y js para el desarrollo de interfaces web responsive design, es decir se adapta al dispositivo sin que el usuario tenga que realizar ninguna acción.

Figura 4 Tecnología Tradicional

Figura 5 Archivos bootstrap

Boostrap nos permite centrarnos en el contenido HTML y dejar de escribir estilos CSS.

2.5.1 Sistema de Grillas.

El sistema de grillas se utiliza para crear diseños de páginas basados en filas y columnas. Boostrap estructura el diseño con el siguiente esquema:

- .col-xs-(numero de columnas): Phones = < 768px
- .col-sm-(numero de columnas): Tablet = > 768px
- .col-md-(numero de columnas): Laptop/Desktop = > 992px
- .col-lg-(numero de columnas): Large Desktop = > 1200px

Al usar esta estructura, bootstrap permite adaptar el diseño de la interface a la pantalla ocultando o mostrando diferentes columnas.

2.5.2 Componentes

Bootstrap ofrece componentes reutilizables desarrollados para proporcionar la iconografía, menús desplegables, grupos de entrada, navegación, alertas entre otros.

Uno de los más importantes de destacar son los glyphicon

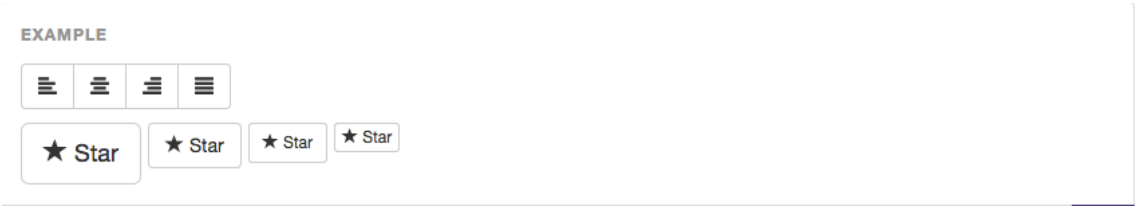


Figura 6 Componentes

Los botones traen una gamma de presentaciones la clase definida para, existen muchos componentes con estilo definido listos para usarse

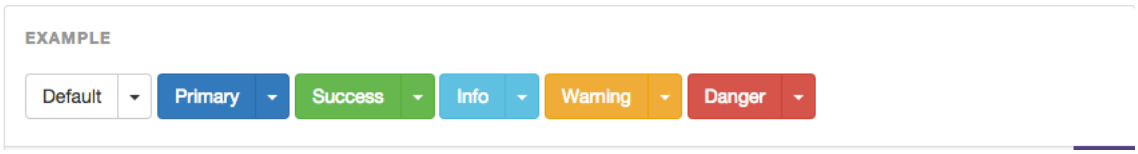


Figura 7 Botones

2.5.3 Lista de componentes

Boostrap ofrece los siguientes componentes:

- Dropdowns
- Buttons
- ButtonsDropDowns
- Nav
- Input
- NavBar

CAPÍTULO 3:

ANÁLISIS Y DISEÑO

3.1 Historias de Usuario

3.1.1 Introducción

El propósito es definir y documentar los requerimientos del Sistema siguiendo las directrices establecidas por una metodología ágil por medio de Historias de Usuario en forma clara, precisa completa y verificable. En este procedimiento se debe identificar todos los requisitos que contendrá el nuevo sistema, esto se logra mediante la aplicación de conversaciones con el cliente a partir de lo cual se analiza e identifican necesidades del producto.

Los requerimientos identificados en esta etapa son validados y de ser el caso evolucionaran a medida que se desarrolle el Sistema, de esta manera se formaliza funciones del sistema junto al cliente.

3.1.2 Alcance

El sistema permitirá optimizar el Proceso de planificación de presupuesto y el proceso de plan de compras de los Laboratorios de Agro calidad, a continuación se describe la función de cada uno de los módulos inmersos en el desarrollo.

3.1.3 Módulo de Usuarios

El sistema cuenta con una vista de autentificación de usuarios donde se ingresa el nombre de usuario registrado y contraseña. Estos serán verificados contra los ya definidos en la base de datos y se determinan los permisos de este usuario en el sistema y el laboratorio al que pertenece. Además un usuario administrador puede agregar nuevos usuarios y administrar los existentes asignándoles su laboratorio y determinados permisos que tendrá luego de autenticarse en el sistema en dependencia de su rol en la empresa.

3.1.4 Módulo de Administración

Este módulo está encargado de definir la parametrización del sistema, es decir un usuario administrador podrá definir y administrar las entidades del sistema de interés de todos los módulos, esto incluye todo lo referente a seguridad como usuarios y roles. Además las entidades de interés general para los laboratorios de Agrocalidad como la definición de los parámetros de los mismos y el área a la que pertenecen, además de los insumos y los procesos a los que pertenecen estos , unidades de medida, enfermedades y sus tipos, proveedores.

3.1.5 Módulo de Laboratorios

Este módulo es utilizado por el personal del laboratorio para definir los servicios prestados por los mismos, los usuarios de este módulo definen en primera instancia los servicios prestados con sus parámetros para luego agregar las diferentes pruebas a realizar como parte de estos servicios en las cuales se introducen los parámetros e indicadores necesarios para el cálculo de presupuesto requerido por la gerencia de Agrocalidad.

3.1.6 Módulo de presupuestos

Este es el módulo principal del sistema en el cual, partiendo de la información proporcionada por el personal de cada laboratorio referente a los precios de los servicios prestados por los laboratorios, la gerencia de Agrocalidad será capaz de hacer un cálculo de presupuesto por un período.

La gerencia será capaz de calcular de una forma clara, rápida e intuitiva los presupuestos para los diferentes periodos pudiéndolos imprimir y llevando una base de datos histórica para posteriores análisis.

3.1.7 Dashboard

Esta pantalla permite brindar un resumen gráfico de los indicadores importantes para la gerencia ayudándole a tener una idea más clara del estado en el que se encuentra tanto la parametrización del sistema como el cálculo del presupuesto.

El gráfico debe mostrar un resumen el total de ensayos parametrizados, el presupuesto calculado para el período y la cantidad total de usuarios registrados en el sistema, además un resumen del costo de cada ensayo registrado.

3.1.8 Visión general del documento

El siguiente capítulo, la sección Descripción general del sistema, en el que se describe de manera resumida la funcionalidad del producto; describiendo los requisitos y se siendo de ayuda estableciendo un contexto para la especificación de requisitos técnicos en el capítulo siguiente. Luego: la sección Especificación de Requisitos, los cuales están escritos

principalmente como guía para el desarrollo y describe en términos técnicos los detalles de la funcionalidad del producto. Ambas secciones del documento describen el mismo producto de software en su totalidad.

3.1.9 Descripción General

El sistema parte de la definición de los diferentes laboratorios y el área al que pertenece, la relación de insumos con sus costos y proveedores; y de los servicios prestados por los diferentes laboratorios apoyados por información previamente establecida en Agrocalidad.

El sistema pretende que mediante el ingreso de costos para los diferentes insumos, el desglose de los servicios prestados por los laboratorios y de los ensayos realizados como parte de la prestación de estos lograr una base de datos capaz de permitir a la gerencia de Agrocalidad hacer un cálculo efectivo del presupuesto para un periodo de tiempo y obtener un plan de compras a realizar por laboratorio para cubrir las necesidades de insumos en el transcurso del mismo.

Técnicamente el producto deberá ser escalable, soporte la inclusión de nuevas funcionalidades y ejecutable en cualquier sistema operativo, a continuación se identificará los usuarios que van a usar el producto, además de la responsabilidad de cada uno de ellos en el sistema.

Para su implementación se cuenta con infraestructura propia, como lo es un servidor blade, una dirección física disponible, y un departamento dispuesto colaborar para la implantación del mismo.

Los módulos del sistema se muestran en la siguiente figura.

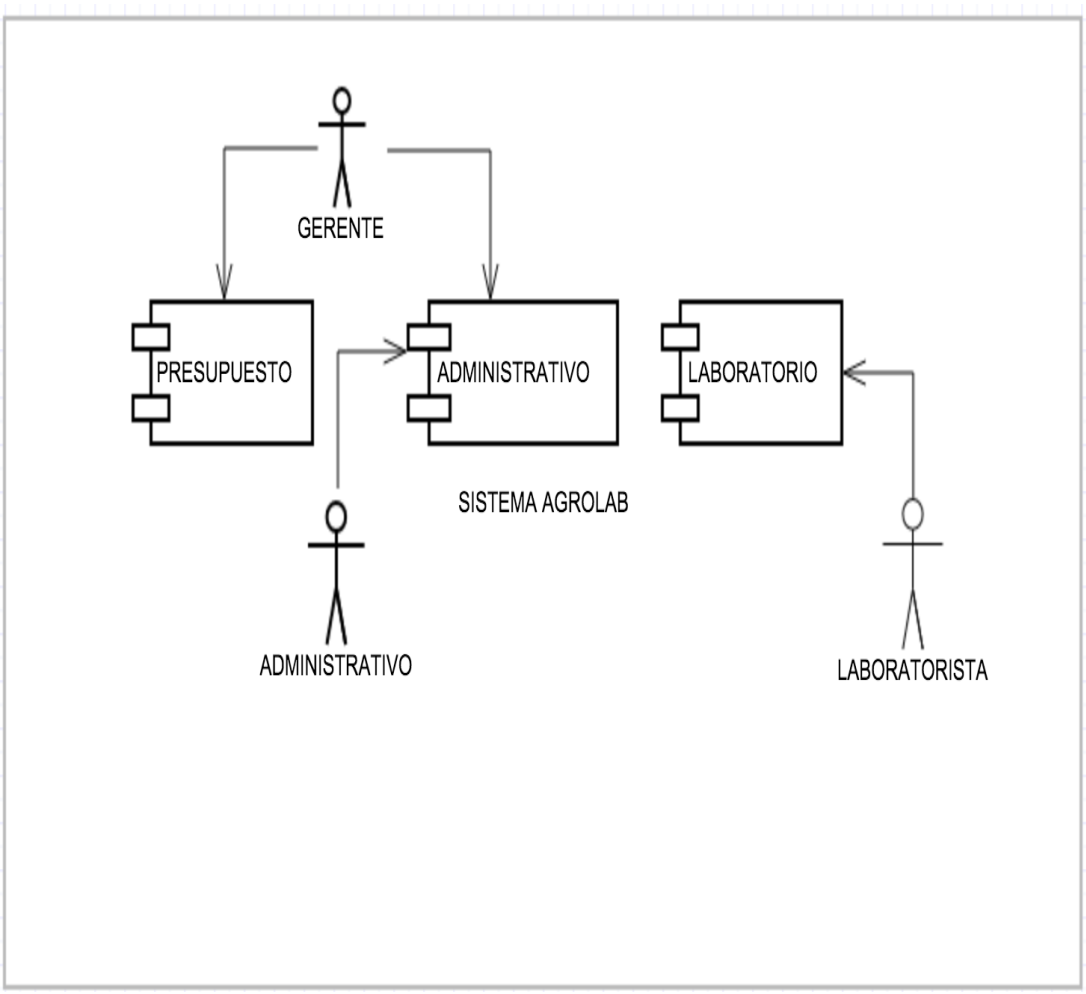


Figura 8 Módulos del sistema

Los usuarios identificados son:

Empleado Administrativo: Son los encargados de registrar los costos de los insumos, los laboratorios y el área al que pertenecen los laboratorios, realizan el proceso de plan de compras. Además mantienen el registro de unidades de medida, proveedores, tipos de enfermedades y enfermedades.

Empleado de Laboratorio: Es el responsable de registrar los servicios que presta el laboratorio. Además realiza el registro y parametrización de los

ensayos que le sirven de base al gerente en el cálculo del presupuesto y plan de compras.

Gerente: Es el encargado del cálculo de los presupuestos y planes de compra para cada período, así como de la supervisión del trabajo de los empleados administrativos e indirectamente de los empleados de cada laboratorio.

3.2 Especificación de Requerimientos Funcionales

En esta sección se describen las historias de usuario del sistema. El gerente, el empleado del laboratorio y el empleado administrativo tienen múltiples funcionalidades cada uno, mientras que el gerente es el actor principal en este sistema, ya que es el que debe proyectar gastos y conformar el presupuesto, los demás actores son fundamentales ya que alimentaran el sistema con información, de insumos, presentaciones, laboratorios, cada uno con su perfil de usuario podrá acceder a sus respectivas funcionalidades.

3.2.1 Historias de Usuario del Gerente

3.2.2 Historia de Usuario: Crear Presupuesto

Breve descripción: El gerente ingresa al sistema, crea un nuevo presupuesto para el período dado apareciéndole inicialmente las cantidades de ensayos a realizar en 0. Permitiéndose editar estas cantidades visualizando constantemente el total de presupuesto y plan de compras.

El plan de compras representa la cantidad de insumos a comprar por laboratorio, por área , obteniendo un reporte en Excel con esta información.

A continuación se realiza la creación de las historias de usuario según nuestra metodología.

Tabla 5

Historia de Usuario Crear Presupuesto

Historia	Crear Presupuesto
ID	AG001
Descripción	Como un
Rol	Gerente
Funcionalidad	**necesito** crear un presupuesto para cada período
Resultado	**con el fin** calcular valores totales
Importancia	800
Criterio de Aceptación	Cálculo de Valores del presupuesto en un período

La tabla es una plantilla de campos compuestos que sirven al desarrollador para evaluar la funcionalidad general del módulo.

3.2.3 Historia de Usuario: Archivo de Presupuesto:

Breve descripción: El gerente ingresa al sistema, selecciona un presupuesto activo y una vez comprobado que está completo puede ejecutar la acción de archivar el mismo de manera tal que este quede registrado en el sistema para posteriores reportes y análisis.

El gerente puede reabrir el archivo si lo requiere, optando por la funcionalidad de editar.

Tabla 6

Historia de Usuario Archivo de Presupuesto

Historia	Archivo de Presupuesto
ID	AG002
Descripción	Como un
Rol	Gerente
Funcionalidad	necesito archivar un presupuesto para cada período
Resultado	con el fin de registrar en la base de datos
Importancia	800
Criterio de Aceptación	Se podrá revisar el presupuesto archivado.

3.2.4 Historia de Usuario: Editar Presupuesto:

Breve Descripción: El gerente ingresa al sistema, selecciona un presupuesto activo y podrá editar la cantidad de cada ensayo a realizar en el período correspondiente. De esta forma se mostrará, basado en el costo de los ensayos el total de presupuesto necesario para realizarlos y el plan de compras correspondiente. De manera tal que el gerente pueda ir ajustando la cantidad idónea para cumplir los objetivos del período.

El sistema le permite al gerente con esta funcionalidad ajustar posibles cantidades para ejecutar su presupuesto.

Tabla 7

Historia de Usuario Editar Presupuesto

Historia	Editar Presupuesto
ID	AG003
Descripción	Como un
Rol	Gerente
Funcionalidad	**necesito** editar la cantidad de ensayos
Resultado	**con el fin** de ajustar el presupuesto para el período
Importancia	800
Criterio de Aceptación	Ajuste de valores al presupuesto.

Para usar esta funcionalidad, se deben implementar las historias de usuario de los demás usuarios, laboratorista, administrativo.

El sistema cuenta con un dashboard de información inicial, en el cual se indicara la ejecución del presupuesto y la parametrización de la misma.

3.2.5 Historia de Usuario: Exportar Presupuesto:

Breve Descripción: El gerente ingresa al sistema, selecciona un presupuesto archivado y este podrá ser exportado a un archivo Excel. De manera tal que este pueda ser utilizado como informe de presupuesto y ejecución de plan de compras del período.

Tabla 8

Historia de Usuario Exportar Presupuesto

Historia	Exportar Presupuesto
ID	AG004
Descripción	Como un
Rol	Gerente
Funcionalidad	necesito la información del presupuesto en un archivo en Excel
Resultado	**con el fin** de presentar informes
Importancia	800
Criterio de Aceptación	Descargar el archivo en formato Excel

3.2.6 Historia de Usuario: Reabrir Presupuesto

Breve Descripción: El gerente ingresa al sistema, selecciona un presupuesto archivado y lo vuelve a abrir de manera tal que le permita editar de nuevo el presupuesto en caso de errores o de algún ajuste al mismo.

Tabla 9

Historia de Usuario Exportar Presupuesto

Historia	Reabrir Presupuesto
ID	AG005
Descripción	Como un
Rol	Gerente
Funcionalidad	**necesito** reabrir presupuesto
Resultado	**Con el fin** realizar ajustes a un presupuesto archivado.
Importancia	500
Criterio de Aceptación	ajuste de valores al presupuesto archivado

3.2.7 Historia de Usuario: Indicadores

Breve Descripción: El gerente ingresa al sistema, e inmediatamente podrá visualizar, el valor del presupuesto ejecutado, los ensayos ingresados al sistema, los usuarios registrados en el sistema, el porcentaje del presupuesto asignado a cada laboratorio y el costo de los ensayos que se realizan.

En esta historia de usuario se decide mostrar información global del sistema, para ello se utiliza conceptos de dashboard, lo que hace al sistema atractivo, con lo cual se cumple la premisa de obtener una aplicación con interface rica.

Tabla 10

Historia de Usuario Indicadores

Historia	Indicadores
ID	AG006
Descripción	Como un
Rol	Gerente
Funcionalidad	**necesito** indicadores de presupuestos, ensayos
Resultado	**Con el fin** de analizar estos indicadores.
Importancia	500
Criterio de Aceptación	dashboard.

3.2.8 Historias de Empleado Administrativo

3.2.9 Historia de Usuario: Administrar Área

Breve Descripción

El empleado administrativo ingresa al sistema, accede a la pantalla del listado de Áreas y podrá ser capaz de listar, agregar, editar y eliminar un área de manera tal que esta pueda ser configurada en la creación y edición de los laboratorios que pertenecen a esta.

El objetivo principal de esto, junto a la edición es mantener un listado de áreas que se corresponda exactamente con las existentes en la empresa.

Tabla 11

Historia de Usuario Indicadores

Historia	Agregar área
ID	AG007
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** agregar un área
Resultado	**con el fin** de editar , guardar o eliminar
Importancia	100
Criterio de Aceptación	mostrar lista de áreas

3.2.10 Historia de Usuario: Administrar Proveedores

Breve Descripción

El empleado administrativo podrá listar, agregar, editar y eliminar proveedores en sus campos nombre y ruc, de tal manera que se pueda registrar luego los insumos de cada proveedor para posteriores reportes, consultas de información y planes de compras.

Tabla 12

Historia de Usuario Administrar proveedores

Historia	Administrar proveedores
ID	AG008
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar proveedores
Resultado	**con el fin** de editar , guardar o eliminar el proveedor
Importancia	100
Criterio de Aceptación	mostrar lista de proveedores

3.2.11 Historia de Usuario: Administrar Unidades

Breve Descripción

El empleado administrativo podrá listar, agregar, editar y eliminar unidades de medida, de tal manera que se pueda registrar luego los insumos.

Tabla 13 Historia de Usuario Administrar unidades

Historia	Administrar unidades
ID	AG009
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar unidades
Resultado	**con el fin** de utilizarlos como medida
Importancia	100
Criterio de Aceptación	mostrar lista unidades

3.2.12 Historia de Usuario: Administrar Clasificaciones

El empleado administrativo podrá listar, agregar, editar y eliminar clasificaciones, de tal manera que se puedan utilizar en los procesos.

Tabla 14

Historia de Usuario Administrar clasificaciones

Historia	Administrar clasificaciones
ID	AG009
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar clasificaciones
Resultado	**Con el fin** de editar, guardar o eliminar las clasificaciones.
Importancia	100
Criterio de Aceptación	mostrar lista unidades

3.2.13 Historia de Usuario: Administrar Proceso

El empleado administrativo podrá listar, agregar, editar y eliminar procesos, de tal manera que se puedan utilizar en la creación y edición de insumos. Para eliminar se deberá tener en cuenta que está relacionada con clasificaciones.

Tabla 15

Historia de Usuario Administrar proceso

Historia	Administrar proceso
ID	AG010
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar procesos
Resultado	**Con el fin** de editar, guardar o eliminar el proceso.
Importancia	100
Criterio de Aceptación	mostrar lista de procesos

3.2.14 Historia de Usuario: Administrar Insumos

El empleado administrativo podrá listar, agregar, editar y eliminar insumos, de tal manera que se puedan utilizar en la creación y edición de ensayos. De cada insumo se ingresa nombre, especificación, proceso al que pertenece, unidad de medida, precio y proveedores. En el caso de necesitar eliminar este no debe estar relacionado en ningún ensayo.

Tabla 16

Historia de Usuario Administrar insumo

Historia	Administrar insumo
ID	AG011
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar insumos
Resultado	**Con el fin** de editar, guardar o eliminar el insumo.
Importancia	100
Criterio de Aceptación	mostrar lista de insumos

3.2.15 Historia de Usuario: Administrar las Presentaciones de un Insumo

El empleado administrativo podrá seleccionar un insumo y listar, agregar, editar y eliminar presentaciones de este, de tal manera que estas puedan ser utilizadas en el cálculo del plan de compras.

Tabla 17

Historia de Usuario Administrar las Presentación de un Insumo

Historia	Administrar las Presentación de un Insumo
ID	AG012
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	necesito listar y agregar la presentación de un insumo
Resultado	Con el fin de editar, guardar o eliminar el insumo.
Importancia	100
Criterio de Aceptación	mostrar lista de insumos

3.2.16 Historia de Usuario: Tipo de Enfermedades

El empleado administrativo podrá listar, agregar, editar y eliminar tipos de enfermedades, de tal manera que se pueda clasificar y aplicar ensayos característicos de las enfermedades por su tipo.

Tabla 18

Historia de Usuario Tipo de Enfermedades

Historia	Tipo de Enfermedades
ID	AG013
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar tipo de enfermedades
Resultado	**Con el fin** de editar, guardar o eliminar el insumo.
Importancia	100
Criterio de Aceptación	mostrar lista de insumos

3.2.17 Historia de Usuario: Enfermedades

El empleado administrativo podrá listar, agregar, editar y eliminar enfermedades, de tal manera que estas puedan ser asociadas a los diferentes servicios y ensayos prestados por el laboratorio para obtener mayor control sobre la información recopilada mediante reportes.

Tabla 19

Historia de Usuario Enfermedades

Historia	Enfermedades
ID	AG013
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar tipo de enfermedades
Resultado	**Con el fin** de editar, guardar o eliminar el insumo.
Importancia	100
Criterio de Aceptación	mostrar lista de enfermedades

3.2.18 Historia de Usuario: Laboratorio

El empleado administrativo podrá listar, agregar, editar y eliminar laboratorios con sus campos nombre y área a la que pertenece, solo se podrá crear un laboratorio cuando exista un área para su relación. Esto nos permitirá tener una relación de los laboratorios pertenecientes a Agrocalidad que serán tomados en cuenta en el cálculo de presupuesto y plan de compras en conjunto a la información recopilada por los trabajadores de estos, relacionada a la relación de insumos necesarios para la realización de los mismos.

Tabla 20

Historia de Usuario Laboratorio

Historia	Laboratorio
ID	AG014
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar laboratorios
Resultado	**Con el fin** de editar, guardar o eliminar el laboratorio.
Importancia	100
Criterio de Aceptación	mostrar lista de laboratorios

3.2.19 Historia de Usuario: Usuario

El empleado administrativo podrá listar, agregar, editar y eliminar los usuarios pertenecientes a cada laboratorio con sus nombre y apellidos, cedula, nombre de usuario, y contraseña de acceso al sistema, así como los roles a los que pertenece, estableciéndose de esta forma el nivel de acceso que tendrá este usuario y sus datos personales, de manera tal que este pueda entrar al sistema y registrar la información necesaria para la operación del mismo.

Tabla 21

Historia de Usuario Usuario

Historia	Usuario
ID	AG015
Descripción	Como un
Rol	empleado administrativo
Funcionalidad	**necesito** listar y agregar usuarios
Resultado	**Con el fin** de editar, guardar o eliminar el usuario.
Importancia	100
Criterio de Aceptación	mostrar lista de usuarios

3.2.20 Historias de Usuario de los empleados del laboratorio

Historia de Usuario: Servicios.

El empleado del laboratorio podrá listar, agregar, editar y eliminar servicios que presta el laboratorio al que él pertenece manteniendo una relación actualizada de los mismos. Estos servicios son los que se listaran a este empleado cuando realice la modificación o adición de un nuevo ensayo indicando que este ensayo se realiza como parte del servicio en cuestión.

Tabla 22

Historia de Usuario Servicios

Historia	Servicios
ID	AG016
Descripción	Como un
Rol	empleado de laboratorio
Funcionalidad	**necesito** listar y agregar servicios que ofrece el laboratorio
Resultado	**Con el fin** de editar, guardar o eliminar servicios.
Importancia	100
Criterio de Aceptación	mostrar lista de servicios

3.2.21 Historia de Usuario: Ensayo

El empleado del laboratorio podrá listar, agregar, editar y eliminar ensayos con los campos nombre y servicios para los cuales es realizado dicho ensayo.

Se espera además que los empleados del laboratorio tengan experiencia en la preparación de las matrices del plan de compra de los laboratorios lo cual les ayudara a hacer una mejor correspondencia entre su trabajo diario y el sistema.

Tabla 23

Historia de Usuario Ensayo

Historia	Ensayo
ID	AG017
Descripción	Como un
Rol	empleado de laboratorio
Funcionalidad	**necesito** listar y agregar ensayos
Resultado	**Con el fin** de editar, guardar o eliminar el ensayo.
Importancia	100
Criterio de Aceptación	mostrar lista de ensayos

3.3 Requerimientos no funcionales

El sistema estará en línea en una cuchilla del servidor blade de la institución y se accederá a través de la intranet, el sistema está preparado para funcionar en varias plataformas como Linux o Windows ya dependerá de la institución alojar el sistema, el sistema debe ser capaz de procesar 10 transacciones por segundo, toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos,

Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos, la nueva aplicación debe manejar fuentes del alfabeto en español.

El sistema debe poseer interfaces gráficas bien formadas y asegurar que los datos estén protegidos del acceso no autorizado, el sistema será desarrollado por módulos, de tal forma que estos módulos puedan ser modificados a futuro dependiendo.

3.4 CRC

Tabla 24

CRC Budget

Budget	
Responsabilidad	Colaboradores
Iniciar presupuesto	Budget Detail
Representar la información del presupuesto	
Almacenar el presupuesto	

Tabla 25

CRC Budget detail

Budget Detail	
Responsabilidad	Colaboradores
Realizar el cálculo del presupuesto	Budget
Realizar operaciones en el presupuesto	

Tabla 26

CRC Test

Test	
Responsabilidad	Colaboradores
Permitir al usuario ingresar ensayos	Detail Test
Realizar operaciones sobre el ensayo	

Tabla 27

CRC Detail Test

Detail Test	
Responsabilidad	Colaboradores
Realizar la composición del Ensayo	Test,Input
Realizar operaciones sobre la composición del ensayo	

Tabla 28

CRC Input

Input	
Responsabilidad	Colaboradores
Permitir al usuario el ingreso de insumos	Input Unit Type,input type, Laboratory
Realizar operaciones sobre el insumo	

Tabla 29

Input Unit Type

Input Unit Type	
Responsabilidad	Colaboradores
Permitir al usuario ingresar métricas para el insumo	Input
Gestionar operaciones referente a las métricas del insumo	

Tabla 30

Input Unit Type

Clasificación	
Responsabilidad	Colaboradores
Permite al usuario clasificar el proceso al que pertenece el insumo	Input Type

Tabla 31

Laboratory

Laboratory	
Responsabilidad	Colaboradores
Permitir al usuario ingresar laboratorios	Área

Tabla 32

Laboratory Service

Laboratory Service	
Responsabilidad	Colaboradores
Brindar información de servicios que ofrece e laboratorio	Disease

Tabla 33

Disease

Disease	
Responsabilidad	Colaboradores
Almacenar enfermedades	Type Disease
Permitir operaciones con enfermedad	

Tabla 34

Type Disease

Type Disease	
Responsabilidad	Colaboradores
Almacenar tipos de enfermedades	Disease

Tabla 35

Presentación

Presentación	
Responsabilidad	Colaboradores
Permite operaciones sobre la presentación del insumo	Input

Tabla 36

SetviceTest

Service Test	
Responsabilidad	Colaboradores
Permite operaciones sobre el servicio brindado por el laboratorio	Laboratory service

3.5 Diseño de pantallas.

Para el diseño de las pantallas se escogió una plantilla con las siguientes áreas. El diseño adaptativo más conocido como Responsive design para nuestras pantallas será visualmente atractivo.

3.5.1 Pantalla Inicial del Sistema.

La pantalla de login es la pantalla inicial del sistema Agrolab, seguida del dashboard.

3.6 Definición de pantallas

3.6.1 Login

La pantalla de login tendrá los campos de nombre de usuario y contraseña.

Bootstrap se encargara de el estilo de la interface usando sus componentes acompañado de font awesome nos dará como resultado una interface agradable.

Figura 9 Login

3.6.2 Barra de Navegación

La barra de Navegación solo permitirá ver el icono del usuario.

Figura 10 Head Bar

3.6.3 Sidebar Izquierdo

Contiene la navegación a los módulos del sistema.

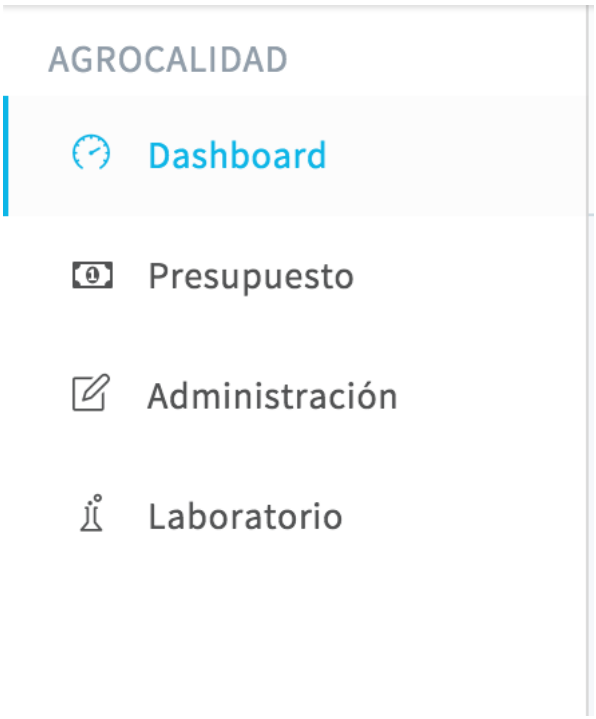


Figura 11 Slidebar

3.6.4 SideBar Derecho

Contiene información del usuario, la funcionalidad de editar usuario y salir del sistema

Figura 12 Slidebar

3.6.5 Región Central

Esta región contendrá la tabla de datos y los botones para su respectiva funcionalidad.

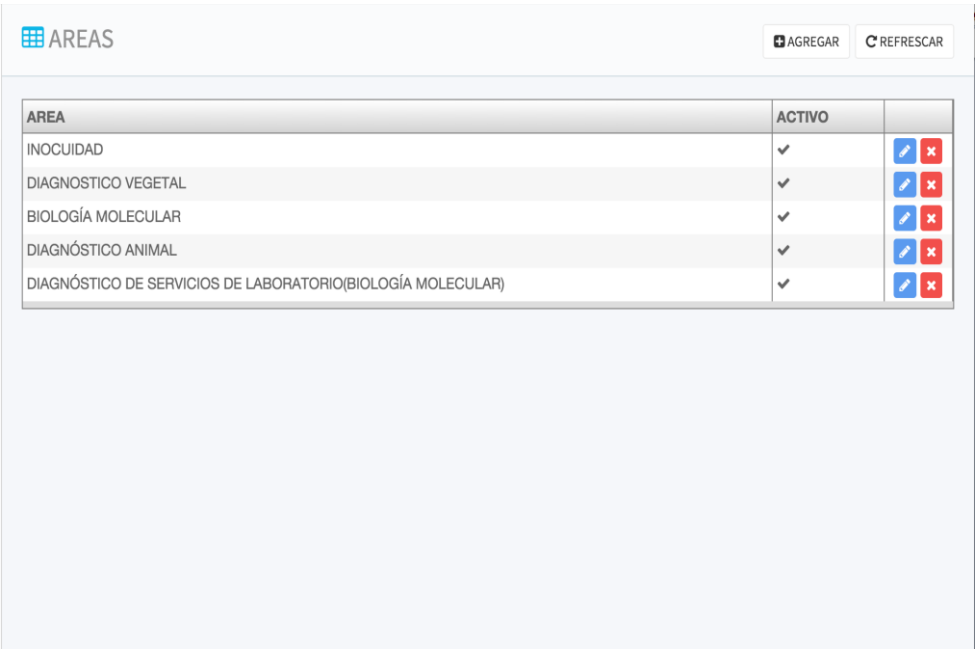


Figura 13 Center

3.6.6 Footer

Esta región contendrá el pie de la página.

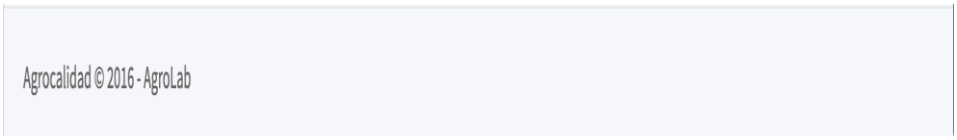


Figura 14 Footer

3.7 Estándar de Codificación

El marco de trabajo es utilizado en proyectos con requisitos funcionales cambiantes, su desarrollo se lo realiza mediante iteraciones (Sprints), cada iteración no debe superar las 2 semanas y debe entregarse un producto funcional al finalizar cada una. Además se realizan reuniones diarias con una duración promedio de 15 minutos, con la finalidad de retroalimentarse con

los problemas y soluciones presentadas en el transcurso de la iteración finalizada y la coordinación e integración del producto a entregar.

La siguiente tabla se enfoca en los estándares.

Tabla 37

Estándar

Identificador	Reglas	Ejemplo
Paquetes	El prefijo del nombre de un paquete se escribe siempre en minúsculas, se empieza utilizando nombres de dominios de alto nivel.	ec.com.agrolab.entities
Clases	Los nombres de las clases deben ser sustantivos, regla camelcase	BudgetEntry AreaController
Interfaces	Siguen la misma regla de las Clases con la característica que llevaran la palabra repository	AreaRepository
Constantes	Deben ir en mayúsculas separando las palabras con un guión bajo.	SPRING_SECURITY_LAST_EXCEPTION

CAPÍTULO 4:

DESARROLLO DEL SISTEMA

Para iniciar el desarrollo del proyecto se ha propuesto realizar un sprint dedicado al levantamiento de requerimientos para determinar su alcance y la planificación a seguir

4.1 SPRINT 0: Planificación de sprints

En esta etapa se determina cuáles son los requisitos funcionales y no funcionales del sistema, los recursos necesarios y el producto final que se obtendrá.

4.2 Lista del Producto

En la lista representaremos la visión del cliente respecto a la funcionalidad requerida por el cliente, este documento estará en evolución mediante avance cada iteración y se han dividido las historias de usuario, a continuación se presenta una lista priorizada para la planificación de los Sprint, agrupando de forma coherente con el fin de presentar un producto final.

Una Lista de Producto nunca está completa por motivos de adaptación al cambio, la lista solo refleja los requisitos concisos y mejor entendidos al principio. La Lista de producto es dinámica, cambia constantemente en función de la necesidad, calidad y funcionamiento.

La Lista de Producto inicial enumera todas las características, funcionalidades, requisitos del producto a entregar, las historias de usuario presentadas en la lista del producto poseen funcionalidades macro del sistema.

4.3 Planificación de Sprints

En primer lugar detallar que la metodología a seguir se basa en las metodologías ágiles que hemos descrito en este capítulo, como lo es scrum y siguiendo el orden predefinido para la tesis, describiremos la ejecución del proyecto con los artefactos definidos por la metodología que realmente nos resulten útiles buscando el objetivo principal de las metodologías ágiles, no utilizar una metodología rígida sino adaptarla a nuestras necesidades.

Para afrontar la ejecución del desarrollo se han identificado los objetivos de negocio en historias de usuario, asignado un coste aproximado (C.E.), un valor de negocio (V.A) y priorizando (P) en función de estos valores. Para estimar el coste no se han utilizado técnicas específicas como las mencionadas en la descripción de la metodología, debido a que el equipo se compone de una única persona y las ventajas que nos ofrecen son fomentar la participación y evitar los malentendidos entre los miembros de un equipo. Para dar prioridad hemos tenido en cuenta nuestros objetivos de negocio.

Otro punto a diferenciar entre Scrum y nuestra ejecución son las reuniones. Aunque se han celebrado reuniones después de cada iteración (reunión de revisión y de preparación) y se ha ido revisando avance del sprint durante su ejecución, tenemos en cuenta que disponemos de un tiempo limitado y para obtener un producto con una funcionalidad aceptable los sprints no deben superar la duración de 2 semanas. A partir de la

duración de 2 semanas por sprint hemos asignado coste estimado en el Product Backlog y la velocidad media de desarrollo debería ser de 1200 puntos por semana.

Dada la planificación anterior podemos definir los sprints agrupando historias de usuario según su prioridad. La siguiente tabla nos muestra las historias de usuario priorizadas y agrupadas por iteraciones hasta la iteración 2 que forman la versión reléase.

A continuación se detalla los sprints que forman parte del proyecto, según la metodología adoptada.

Tabla 38

Back Log

Identificador	Descripción	Coste Estimado	Condiciones de satisfacción	Valor Aportado
AG007	Creación de Área	400	Quiero crear el área a través de un formulario.	1000
AG007	Modificar Área	100	Podemos editar un area.	50
AG007	Eliminar Area	100	Podemos eliminar un area	100
AG014	Creación de Laboratorio	400	Quiero crear laboratorio a través de un formulario.	100
AG014	Modificar Laboratorio	100	Se podrán modificar los datos del laboratorio.	1100
AG014	Eliminar Laboratorio	100	Al acceder a una historia de usuario puede eliminar el laboratorio	1100
AG016	Creación de Servicios	400	Quiero crear servicios que ofrece el laboratorio a través de un formulario	200
AG016	Modificación de Servicios	100	Se podrán modificar los datos del servicio.	1100
AG016	Eliminación de Servicios	100	Podemos eliminar un servicio	1000

AG013	Creación de Enfermedades	400	Quiero crear la enfermedad a través de un formulario	800
AG013	Modificación de Enfermedades	100	Se podrán modificar los datos de la enfermedad.	100
AG013	Eliminación de Servicios	100	Podemos eliminar un servicio	100
AG015	Creación de usuarios	400	Quiero crear el usuario a través de un formulario.	1000
AG015	Modificar usuario	100	Se podrán modificar los datos de un usuario.	100
AG015	Eliminar usuario	100	Podemos eliminar un usuario	1100
AG018	Creación de tipo de Enfermedad	400	Se podrán modificar los datos de la enfermedad.	1400
AG018	Modificación de tipo de Enfermedad	100	Se podrán modificar los datos de la enfermedad.	1100
AG018	Eliminación de tipo de Enfermedad	100	Se podrán modificar los datos de la enfermedad.	100
AG010	Creación de Proceso	400	Quiero crear un proceso a través de un formulario.	200
AG010	Modificar Proceso	100	Se podrán modificar los datos del proceso.	100
AG010	**Eliminar Proceso**	**100**	**Podemos eliminar un proceso.**	50

Identificador Objetivo	Descripción	Coste estimado	Condiciones de satisfacción	Valor aportado
AG09	Creación de Tipo de unidad de insumo	400	Quiero crear el usuario a través de un formulario.	800
AG09	Modificar Tipo de unidad de insumo	100	Podemos editar un tipo de unidad de insumo.	100
AG09	Eliminar Tipo de unidad de insumo	100	Podemos eliminar un tipo de unidad de insumo	100
AG11	Creación de insumo	400	Se podrán modificar los datos de insumo.	400
AG11	Modificación de insumo	100	Se podrán modificar los datos de insumo.	100
AG11	Eliminación de insumo	100	Podemos eliminar un insumo.	100
AG12	Creación de Presentación	400	Quiero crear la presentación de insumos a través de un formulario.	400
AG12	Modificación de Presentación	100	Se podrán modificar los datos de la presentación.	100
AG12	Eliminar Presentación	100	Al acceder a una historia de usuario puede eliminar la presentación	200
AG16	**Creación de Ensayo**	**400**	**Quiero crear un ensayo que ofrece el laboratorio a través de un formulario.**	100

Identificador Objetivo	Descripción	Coste estimado	Condiciones de satisfacción	Valor aportado
AG01	Crear Presupuesto	1000	Quiero crear un presupuesto a través de un formulario.	1400
AG01	Modificar presupuesto	500	Podemos editar un presupuesto.	1100
AG01	Eliminar presupuesto	500	Podemos eliminar un presupuesto	1100
AG02	Archivar presupuesto	500	Se podrán archivar los datos del presupuesto.	1400
AG02	Exportar presupuesto	1000	Se podrán exportar los datos del presupuesto.	1100
AG04	Modificar presupuesto	500	Se podrán modificar el presupuesto.	100
TOTAL		**10600**		

Tabla 39 Sprint 1

Sprint	id	Descripción	C.E	V.A	P
	AG07	Modificar Area	100	50	**2**
	AG07	Eliminar Area	100	100	**3**
	AG14	Creación de Laboratorio	400	100	**4**
	AG14	Modificar Laboratorio	100	1100	**5**
	AG14	Eliminar Laboratorio	100	1100	**6**
	AG16	Creación de Servicios	400	200	**7**
	AG16	Modificación de Servicios	100	1100	**8**
	AG16	Eliminación de Servicios	100	1000	**9**
	AG13	Creación de Enfermedades	400	800	**10**
	AG13	Modificación de Enfermedad	100	100	**11**
	AG13	Eliminación de Servicios	100	100	**12**
	total		**2200**		

Tabla 40

Sprint 2

Sprint	id	Descripción	C.E	V.A	P
	AG015	Modificar usuario	100	100	**14**
	AG015	Eliminar usuario	100	110	**15**
	AG018	Creación tipo deEnfermedad	400	140	**16**
	AG018	Modificación tipo	100	110	**17**
	AG018	Eliminación tipo Enfermedad	100	100	**18**
	AG010	Creacion de Proceso	400	200	**19**
	AG010	Modificar Proceso	100	100	**20**
	AG010	Eliminar Proceso	100	50	**21**
	total		**2200**		

Tabla 41 Sprint 3

Sprint	id	Descripción	C.E	V.A	P
	AG09	Modificar unidades de insumos	100	100	**26**
	AG09	Eliminar unidades de insumos	100	100	**27**
	AG11	Creación tipo deEnfermedad	400	400	**28**
	AG11	Modificación tipo Enfermedad	100	100	**29**
	AG11	Eliminación tipo Enfermedad	100	100	**30**
	AG12	Creacion de Proceso	400	400	**31**
	AG12	Modificar Proceso	100	100	**32**
	AG12	Eliminar Proceso	100	200	**33**
	AG16	Creacion de Tipo de Insumo	400	100	**34**
	AG16	Modificiación de Tipo de	100	100	**35**
	AG16	Eliminación de Tipo de Insumo	100	100	**36**
total		**2200**			

Tabla 42 Sprint 4

Sprint	id	Descripción	C.E	V.A	P
	AG01	Creación de presupuesto			
	AG01	Modificar presupuesto	500	110	**38**
	AG01	Eliminar presupuesto	500	110	**39**
	AG02	Archivar presupuesto	500	140	**40**
	AG03	Exportar presupuesto	100	110	**41**
	AG04	Modificar presupuesto	500	100	**42**
total		**4000**			

4.4 Desarrollo

En esta sección vamos a detallar cómo hemos afrontado el proyecto, la descomposición de las historias de usuario en tareas, entrando en detalle de las tareas que se consideren oportunas y detallando el cambio, describiendo así la organización y metodología seguidas para el desarrollo.

En el Product Backlog se han incluido las historias de usuario funcionales que enumeramos como resultado de los objetivos de negocio.

Cabe destacar que en el Product Backlog no se ha tenido en cuenta la primera fase, la ejecución de la memoria, algunos aspectos de formación e investigación en metodologías ágiles.

Como base para comenzar estas tareas se han tenido en cuenta:

- Escribir qué se debe hacer y no cómo se debe hacer.
- Las tareas deben seguir los principios INVEST.

4.5 Sprint 1

Para la planificación del Sprint se llevó a cabo una reunión con el Product Owner .En esta reunión se realizó un análisis de los requerimientos iníciales, el modelo entidad relación de la base de datos y el diseño de las pantallas de login y los menús a desplegar en la aplicación.

En esta iteración vamos a tratar las siguientes historias de usuario, estas historias se descomponen en el siguiente Sprint Backlog. Como vemos en la tabla, el número de tareas respecto a la iteración anterior ha aumentado considerablemente. Teniendo en cuenta el número de tareas y que existen temas desconocidos para el encargado de las mismas, ampliaremos la duración de esta iteración a 2 semanas.

En este Sprint implementaremos el módulo de laboratorios y sus relaciones.

Tabla 43

Tareas del Sprint 1

Backlog	Tarea	Tipo	Descripción	R	E
AG007, AG014, AG016, AG013	Tarea 1	Análisis ,diseño	Modelo de datos	L.A	30%
AG007, AG014, AG016, AG013	Tarea 2	Implement ación	Crear clases MVC	L.A	20%
AG007, AG014, AG016, AG013	Tarea 3	Implement ación	Creación de formularios y métodos	L.A	30%
AG007, AG014, AG016, AG013	Tarea 4	Implement ación	Aplicar las correcciones visuales en la estructura de las páginas.	L.A	20%

La información de la tabla nos detalla la estimación realizada en este y el resto de sprints backlogs. En este caso y debido al entorno y variabilidad que podemos obtener en las horas dedicadas en un día se ha decidido incluir un porcentaje de la duración completa del Sprint. Dado el tipo de tareas que tratamos en esta iteración.

4.6 Sprint 2

En este Sprint se implementara la funcionalidad de creación de usuarios y entidades referentes al insumo.

Tabla 44

Tareas del Sprint 2

Backlog	Tarea	Tipo	Descripción	R	E
AG015, AG018, AG010, AG019	Tarea 5	Análisis ,diseño	Modelo de datos	L.A	30%
AG015, AG018, AG010, AG019	Tarea 6	Implementación	Crear clases MVC	L.A	20%
AG015, AG018, AG010, AG019	Tarea 7	Implementación	Creación de formularios y métodos	L.A	30%
AG015, AG018, AG010, AG019	Tarea 8	Implementación	Aplicar las correcciones visuales en la estructura de las páginas.	L.A	20%

4.7 Sprint 3

Para la planificación del Sprint1 se llevó a cabo una reunión con el Product Owner. En esta reunión se realizó un análisis de la funcionalidad para el terminar la parametrización de las entidades que intervienen en el proceso macro que es el presupuesto.

Tabla 45

Tareas Sprint 3

Backlog	Tarea	Tipo	Descripción	R	E
AG009, AG011, AG012, AG016	Tarea 9	Análisis, diseño	Modelo de datos	L.A	30%
AG009, AG011, AG012, AG016	Tarea 10	Implementación	Crear clases MVC	L.A	20%
AG009, AG011, AG012, AG016	Tarea 11	Implementación	Creación de formularios y métodos	L.A	30%
AG009, AG011, AG012, AG016	Tarea 12	Implementación	Aplicar las correcciones visuales en la estructura de las páginas.	L.A	20%

4.8 Sprint 4

En este Sprint se encuentra el producto final del sistema, lo que permitirá al usuario sacar la información automatizada del proceso de costos y presupuestos.

Tabla 46

Tareas Sprint 4

Backlog	Tare	Tipo	Descripción	R	E
AG001, AG002, AG004	Tarea 13	Análisis ,diseño	Modelo de datos	L.A	30%
AG001, AG002, AG004	Tarea 14	Implement ación	Crear clases MVC	L.A	20%
AG001, AG002, AG004	Tarea 15	Implement ación	Creación de formularios y métodos	L.A	30%

4.9 TESTING

Los tests automatizados son características que más calidad aporta a un proyecto, en el caso de un lenguaje dinámico como javascript creo que su importancia aumenta enormemente al perder al seguridad que ofrece la comprobación de tipos que realiza el compilador.

En ese sentido, angular está diseñado desde el principio para poder ser fácilmente testeable y para ello cuenta con dos tipos de test diferenciados, test unitarios y test de extremo a extremo.

Para realizar los test unitarios se utiliza Karma y Jasmine, herramientas especializadas para la ejecución de test.

Karma - un corredor de prueba que se adapte a todas nuestras necesidades

Una herramienta de línea de comandos JavaScript que se pueden utilizar para generar un servidor web que carga el código fuente de la aplicación y ejecuta sus pruebas. Puede configurar Karma para funcionar contra una serie de navegadores, que es útil para estar seguro de que su aplicación funciona en todos los navegadores que necesita para apoyar. Karma se ejecuta en la línea de comandos y mostrará los resultados de sus pruebas en la línea de comandos una vez que se ejecute en el navegador

Jazmín es un marco de desarrollo comportamiento impulsado para probar el código JavaScript. No depende de ningún otro frameworks de JavaScript. No requiere un DOM. Y tiene una sintaxis limpia, evidente por lo que se puede escribir fácilmente pruebas.

4.10 Test Unitarios

Con angular podemos escribir test unitarios que nos permitan definir y comprobar el funcionamiento de los componentes de la aplicación por separado

4.11 Estructura de un test

4.11.1 Describe

```
describe("ordenar lista de usuarios", function() {
// test individuales aqui });
```

Se define los test individuales con la llamada a la función.

4.11.2 It

La agrupación de las pruebas relacionadas dentro de describe los bloques y la descripción de cada test individual dentro de una llamada que mantiene su autodocumentado pruebas.

```
describe(ordenar lista de usuarios ', function() {
it(ordena en forma descendente por defecto, function() {
 // your test assertion goes here
 });
});
```

4.11.3 expect

Para completar, Jasmine ofrece comparadores que permiten establecer afirmaciones:

```
describe(ordenar lista de usuarios ', function() {
it(ordena en forma descendente por defecto, function() {
var users = ['jack', 'igor', 'jeff'];
var sorted = sortUsers(users);
expect(sorted).toEqual(['jeff', 'jack', 'igor']);
}); });
```

En la aplicación, los test se enfocaran a probar el acoplamiento de los módulos, se procederá a configurar en archivos js la herramienta karma y jasmine.

```
            'app/agrolab/clasification/clasifications.js',
            'app/js/app.js',
            'app/js/agrolab.js',
            'tests/**/*.spec.js'
        ],

        autoWatch: true,

        frameworks: ['jasmine'],

        browsers: ['Chrome'],

        plugins: [
            'karma-phantomjs-launcher',
            'karma-chrome-launcher',
            'karma-firefox-launcher',
            'karma-jasmine',
            'karma-junit-reporter'
        ],

        junitReporter: {
            outputFile: 'test_out/unit.xml',
            suite: 'unit'
        }

    });
};
```

Figura 15 Karma

A continuación se crean los archivos de configuración de Karma y Jasmine

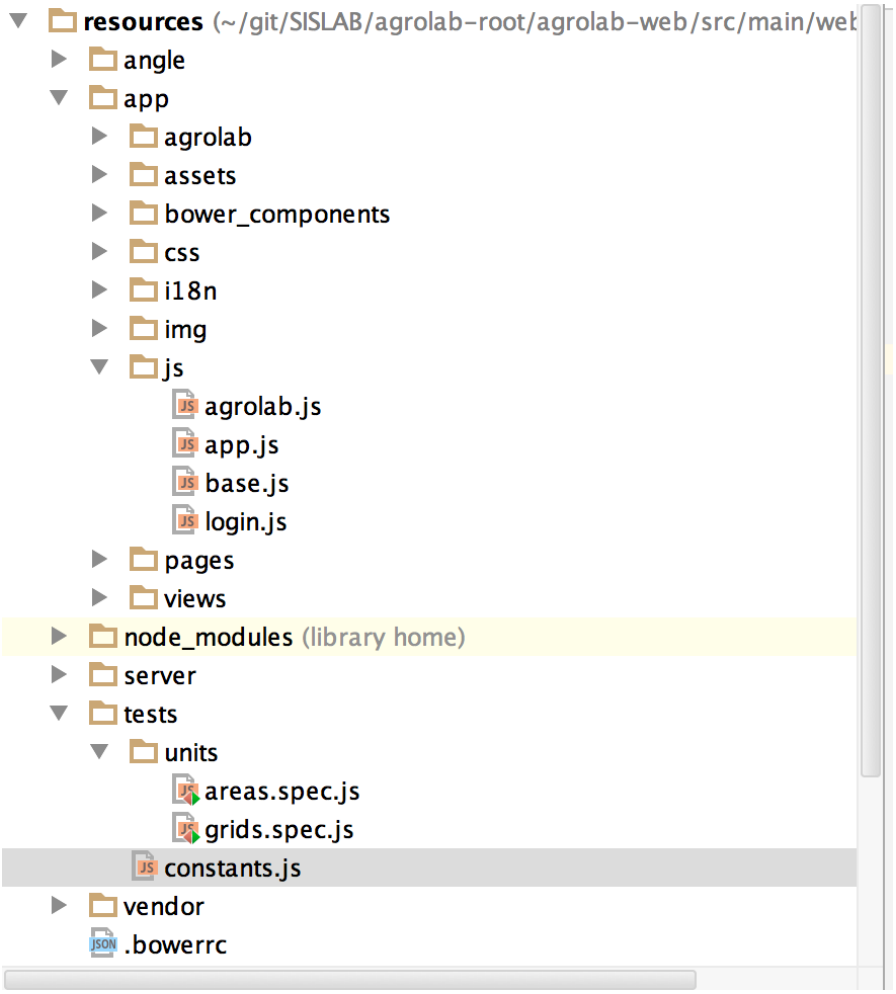


Figura 16 Estructura

Se realiza el test mediante procedimiento establecido en la documentación de Angular JS, como ejemplo se utiliza el controlador de Areas.js y el componente grid que es el central en todos los módulos.

areas.spec.js | grids.spec.js | constants.js | agrolab.js | karma.conf.js | login.html

```
describe('Areas Controller Tests', function () {
    beforeEach(module('AgrolabApp'));
    var AreasCtrl, scope, Grids, Area;

    beforeEach(inject(function ($rootScope, $controller, _Grids_) {
        Grids = _Grids_;
        scope = $rootScope.$new();
    }));

    describe('constructor and inject', function() {
        it('test creation from gridUtils OK', function() {
            beforeEach(inject(function ($rootScope, $controller, _Grids_) {
                Grids = _Grids_;
                scope = $rootScope.$new();
                spyOn(Grids, 'delFunc');
                spyOn(Grids, 'createFunc');
                spyOn(Grids, 'editFunc');
                spyOn(Grids, 'refreshFunc');
                AreasCtrl = $controller('AreasCtrl', { $scope: scope, Grids: Grids});
            }));
            expect(Grids.delFunc).toHaveBeenCalled();
            expect(Grids.createFunc).toHaveBeenCalled();
            expect(Grids.editFunc).toHaveBeenCalled();
            expect(Grids.refreshFunc).toHaveBeenCalled();
        });

        it('test delete call service delete', function() {
            beforeEach(inject(function ($rootScope, $controller, _Grids_, _Area_) {
                Grids = _Grids_;
                Area = _Area_;
                spyOn(Area, 'remove');
                spyOn(Grids, 'delFunc').and.returnValue(function(dato){
```

Figura 17 Test

Resultado

```
Terminal
Local | Local (1)
> karma start karma.conf.js  --single-run

INFO [karma]: Karma v0.12.37 server started at http://localhost:9876/
INFO [launcher]: Starting browser Chrome
INFO [Chrome 48.0.2564 (Mac OS X 10.11.3)]: Connected on socket WDAlak1j0qgsPjqlJYdj with id 44360016
Chrome 48.0.2564 (Mac OS X 10.11.3): Executed 1 of 7 SUCCESS (0.092 secs / 0.088 secs)
MacBook-Pro-de-macintosh:resources macintosh$ npm run test-single-run
```

Figura 18 Resultado

CAPÍTULO 5:

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Las historias de usuario del sistema desarrollado se realizó en forma gradual. A medida que se avanzaba con los diferentes prototipos del sistema, nos permitió realizar correcciones y ajustes al producto de software creado.

- El marco de trabajo SCRUM permitió identificar claramente las necesidades de la Institución mediante las entregas continuas, las modificaciones de requerimientos en cada sprint demostró que se cumple con el concepto de flexibilidad y adaptación.

- AngularJS permite dinamizar nuestras páginas HTML en el lado del cliente.

- Este framework permite separar la lógica de presentación de la vista mediante la vinculación de datos del modelo con componentes HTML.

- La separación de la lógica de presentación de la vista, de forma que el código Javascript es independiente al código HTML por lo que se puede rehacer la vista sin necesidad de tener que tocar ni una sola línea de nuestra lógica de negocio.

5.2 Recomendaciones

- Para la realización de un sistema que permita automatizar los procesos de una institución pública, es recomendable conocer y entender los procesos, para recolectar los requerimientos de una manera más fácil y rápida evitando errores en la fase de desarrollo.

- Para el desarrollo de aplicaciones web es recomendable usar herramientas de software libre, la tendencia actual es la utilización de estas herramientas que aparte de economizar totalmente la construcción de un sistema, garantizan y facilitan en gran manera el desarrollo, a más de que nos permiten reutilizar el código y optimizarlo.

- Para el desarrollo de proyectos pequeños es aconsejable combinar las metodologías agiles Scrum con Xtreme Programming ya que proporciona buenos resultados en cortos plazos, y con un equipo de desarrollo no muy extenso; la utilización de estas dos metodologías serán exitosas siempre y cuando exista una buena interacción con el usuario final.

- Las metodologías ágiles son opuestas a las metodologías predictivas, ofreciendo un enfoque más adecuado para determinados proyectos como el desarrollo de software. La convergencia entre ambos modelos puede dar lugar a una gestión eficiente y eficaz.

- Se recomienda a Agro calidad utilizar el actual proyecto de tesis como una herramienta para la automatización a futuro del proceso de gestión de costos y presupuestos por ensayos.

6 Bibliografía

AngularJS Developer Guide. (01 de 01 de 2010). *Developer Guide*. Recuperado el 01 de 10 de 2015, de https://docs.angularjs.org/guide/controller

Manifiesto Agil. (01 de 01 de 2011). *Manifiesto Agila para el desarrollo de software*. Recuperado el 01 de 10 de 2015, de http://agilemanifesto.org/iso/es/

MIT. (01 de 01 de 2015). *Javascript Bootstrap*. Recuperado el 01 de 10 de 2015, de http://getbootstrap.com/javascript/

Panda, S. (2014). *AngularJS: Novice to Ninja.* California: SitePoint.

Sandoval, J. (2009). *RESTful Java Web Services.* New York: Packt Publishing.

Scrum Agile. (01 de julio de 2013). *Scrum Basics*. Recuperado el 01 de 08 de 2015, de Scrum Basics: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf

Scrum Org. (01 de 01 de 2009). *Scrum Alliance*. Recuperado el 01 de 08 de 2015, de what is Scrum: https://www.scrum.org/resources/what-is-scrum/