



**Facultad de Ingeniería**  
**Bernard Wand Polak**

**Ingeniería de Software en la**  
**Práctica**

**Obligatorio 2**

Facundo Laxalde Nro Est. 174323  
Francisca Galperin Nro Est. 150138  
Lucía d'Avila Nro Est. 180658

Grupo N7A  
Docente: Ignacio Valle

# Índice

<b>Abstract</b>	<b>5</b>
<b>Glosario</b>	<b>6</b>
<b>1. Objetivos</b>	<b>7</b>
<b>2. Tecnologías y herramientas de desarrollo</b>	<b>8</b>
<b>3. Enfoque metodológico</b>	<b>9</b>
3.1 Justificación	9
<b>4. Asignación de roles</b>	<b>11</b>
<b>5. Análisis</b>	<b>12</b>
5.1 El problema	12
5.2 Actores del sistema	13
5.3 Alcance del Producto	13
5.4 Especificación de requerimientos	14
Defensa del primer obligatorio	14
5.5 Cambios en el Alcance	20
5.6 Casos de uso	20
<b>6. Gestión de la comunicación</b>	<b>29</b>
6.1 Plan de comunicación	30
<b>7. Gestión de proyecto</b>	<b>32</b>
7.1 Cronograma	32
Primera entrega: 27/04/2017	32
Reunión con el cliente (Defensa primer entrega): 8/05/2017	32
Segunda entrega: 22/06/2017	32
Demo al Cliente: fecha a definir	32
7.2 Gestión de tareas	33
7.3 Gestión de riesgos	33
7.3.2 Análisis de Riesgos	34
Evolución de los riesgos en el tiempo	35
<b>8. Gestión de la configuración</b>	<b>37</b>
8.1 Control de cambios y de versiones	37
8.1.1 Solicitudes de cambio	37
8.1.2 Versionado	38
8.1.3 Gestión de defectos	38
<b>9. Aseguramiento de la calidad</b>	<b>40</b>
9.1 Proceso y producto	40
9.1.1 Análisis de calidad	40
9.1.1.1 Actividades preventivas	41

9.1.1.2 Actividades correctivas	42
9.1.1.3 Actividades registrales	42
9.1.1.3.1 Registro de Esfuerzos	43
9.1.1.3.2 Estimaciones	43
9.1.2 Plan de métricas	43
9.1.3 Oportunidades de mejora	45
9.2 Pruebas de software	46
9.3.1 Pruebas unitarias	46
9.3.2 Pruebas de aceptación	47
9.3.3 Pruebas de integración	57
9.3 Estándares de documentación	66
<b>10. Arquitectura</b>	<b>67</b>
10.1 Diseño de la solución	67
10.1.1 Arquitectura	67
10.1.1.2 Diagrama de componentes	67
10.1.2 Diseño del FrontEnd	69
Primera entrega	69
Segunda entrega	69
10.1.2.1 Mock-ups	69
10.1.2 Diseño del BackEnd	70
10.1.2.1 Almacenamiento de datos en la aplicación	70
10.1.3 Justificaciones de Diseño	70
10.2 Modelo de Base de datos	71
10.2.1 Persistencia de Datos	71
10.2.1 Modelo de tablas	72
10.2.2 Datos de Prueba	73
<b>11. Implementación</b>	<b>74</b>
11.1 Código	74
11.1.1 Aplicación y evidencia de Clean Code	74
<b>12. Manual de instalación</b>	<b>76</b>
<b>13. Referencias</b>	<b>77</b>
<b>14. Anexos</b>	<b>78</b>
14.1 Actividades registrales	78
14.1.1 Registro de esfuerzos	78
14.1.2 Estimaciones	78
14.2 Mock-ups	80
14.2.1 Menú inicial residente	80
14.2.2 Menú inicial administrador	81
14.2.3 Login residente	82
14.2.4 Login administrador	83

14.2.5 Alta anuncio residente	84
14.2.6 Alta anuncio administrador	85
14.2.7 Control de asistencia a reunión	86
14.2.8 Alta reserva	87
14.2.9 Cierre votación	88

# Abstract

Este trabajo detalla el proceso de construcción de una aplicación móvil en un contexto de elaboración de software. La presente documentación describe tanto la metodología como el proceso que tiene como resultado el producto final: la aplicación VecinosUY. La misma se realiza en respuesta a la consigna establecida en la letra de Obligatorio de la materia Ingeniería de software en la práctica. Dicho proceso se llevará a cabo por tres estudiantes del dictado de la materia.

Las justificaciones pertinentes para cada uno de las partes se encuentran en la parte de Anexos, en donde se encontrará la información detallada de cada una de las actividades realizadas desde el primer día de trabajo.

# Glosario

Framework: “Es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, que puede servir de base para la organización y desarrollo de *software*. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.”

ASP.NET Web API: “Es un marco que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen exploradores y dispositivos móviles. ASP.NET Web API es la plataforma perfecta para crear aplicaciones RESTful en .NET Framework.”

BackEnd: “Refiere a lo que hay del lado del servidor. Es decir, describe básicamente cómo la aplicación trabaja, se actualiza y modifica. Refiere a estos aspectos que el usuario, desde su rol, no puede ver.”

FrontEnd: “Es la parte del software que interactúa con el o los usuarios. Su desarrollo implica dar formato a contenidos, desarrollo del aspecto de la web (interfaz gráfica) y manipular los resultados obtenidos.”

Mock-up: “Es un modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Los mockups son utilizados por los diseñadores principalmente para la adquisición de comentarios por parte de los usuarios. “

Angular JS: “Es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.”

Visual Studio: “Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC.

SQL Server: “Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.”

# 1. Objetivos

Como objetivo nos proponemos la construcción de una aplicación Android de nuestra propia elección que nos permitirá realizar una prueba de concepto para futuros proyectos, enfocándonos principalmente el proyecto de grado. Con este objetivo en mente, formamos el equipo de trabajo que planeamos mantener para el proyecto final.

El desafío se encuentra no solo en una implementación utilizando una tecnología en la que ninguno de los miembros del equipo de proyecto tiene experiencia, sino también en la definición de un proceso y la elección de una metodología que se ajuste a los objetivos del proyecto y al equipo.

## 2. Tecnologías y herramientas de desarrollo

Las tecnologías principales utilizadas para la implementación no fueron decisión del equipo de trabajo, sino dadas como una restricción por parte de la cátedra.

Para el back-end debemos utilizar ASP.NET Web Api con Visual Studio y SQL Server como motor de base de datos. Para el front-end debemos realizar un desarrollo mobile utilizando Android Studio.

Adicionalmente, decidimos realizar una interfaz diferente para los usuarios administradores. Construiremos una página web utilizando Angular JS. La elección de esta tecnología si fue tomada por el equipo y se debe a que todos contamos con experiencia de desarrollo en la misma.

Durante las etapas de implementación y testing utilizamos también GenyMotion como emulador para ejecutar la aplicación ya que el mismo es mucho más robusto y eficiente que emulador incorporado en Android Studio.



## 3. Enfoque metodológico

### 3.1 Justificación

Para la realización de este proyecto decidimos utilizar el enfoque metodológico basado en planes. En primer lugar, y quizás la principal razón por la cual consideramos que un enfoque tradicional era el más indicado, es que el desarrollo del producto se realizará con requerimientos fijos, establecidos y acordados previamente. Esta disciplina tiene a ser cada vez menos usada en la industria del software, un artículo publicado por HP establece que a Febrero del 2016 alrededor de  $\frac{2}{3}$  de las empresas de software se inclina a metodologías ágiles o ya las utiliza completamente (1) debido a que el cambio frecuente en los requerimientos es una característica que identifica a la mayor parte de los proyectos.

Por otro lado, consideramos como factor muy importante al elegir el enfoque basado en planes es el hecho de que tenemos experiencia de varios años utilizando esta metodología, lo que nos juega un papel fundamental a la hora de trabajar.

En este caso, por tratarse de un emprendimiento de innovación donde no contamos con un cliente, el alcance será definido inicialmente y confiamos en que los cambios que surjan durante el transcurso del proyecto serán de bajo impacto y podrán ser gestionados sin poner en riesgo el éxito del proyecto.

Por otra parte, este enfoque nos permite realizar un seguimiento detallado de cada una de las fases que forman parte del ciclo de vida del proyecto. Utilizaremos un ciclo de vida en Cascada. En este enfoque las etapas del proceso se ordenan y definen rigurosamente. El comienzo de cada etapa depende de la finalización de la etapa anterior. Al final de cada etapa, se realiza un análisis donde se determina si el proyecto está listo para avanzar a la siguiente.

Las etapas que forma este ciclo de vida se listan a continuación:

1. **Análisis de Requerimientos:** En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir.
2. **Diseño:** En este proceso se construye un modelo o representación de lo que se va a construir posteriormente y se toman las decisiones necesarias y se definen tácticas para obtener un sistema que cumpla con los atributos de calidad establecidos.

3. **Implementación:** En esta etapa se desarrolla el sistema.
4. **Pruebas:** En esta etapa se comprueba que el sistema funciona correctamente y que cumple con los requisitos definidos.
5. **Mantenimiento:** En esta etapa se lleva a cabo la modificación del producto después de la entrega, para corregir errores, mejorar el rendimiento, u otros atributos

La etapa de Mantenimiento se encuentra fuera del alcance del proyecto.

## 4. Asignación de roles

Identificamos los roles necesario para llevar a cabo el proyecto, la asignación de los mismo entre el equipo de trabajo ayudará a estructurar el mismo, y a crear conciencia de las responsabilidades de cada uno de los miembros.

En este proyecto, al tratarse de un equipo de pocos integrantes, todos participamos de mayor o menor manera en todas las áreas. De todas formas, asignamos los siguientes roles:

<b>Rol definido</b>	<b>Nombre integrante</b>
Responsable de Desarrollo	Facundo Laxalde
Responsable de Gestión del Proyecto	Francisca Galperin
Responsable de Calidad	Lucía d'Avila

A lo largo de este documento, nos referiremos a los roles para definir el encargado de llevar a cabo las tareas mencionadas.

## 5. Análisis

En esta etapa realizamos una serie de tareas con el fin de identificar las necesidades de los usuarios finales y los objetivos que debe cumplir el producto. Identificamos a los actores involucrados en el sistema a desarrollar y las necesidades de cada uno.

Por otra parte, detallaremos el alcance específico del producto y especificaremos en detalle los requerimientos a desarrollar. Con el fin de describir las funcionalidades en detalle, realizamos casos de usos asociados a cada uno de los requerimientos funcionales definidos. Por otro lado, detallamos también los requerimientos no funcionales que deberá cumplir el sistema, eso son restricciones dadas por la cátedra o atributos de calidad que el sistema deberá cumplir.

### 5.1 El problema

Inicialmente pensamos en implementar una idea que involucre la vida diaria de las personas, y además solucione problemas frecuentes de alguna forma sencilla. Eso no es nada sencillo...

A su vez, decidimos apuntar a un público amplio. Es por eso que luego de varias discusiones, definimos el problema donde apuntaríamos: mejorar la comunicación entre los residentes de los edificios, algo bastante común de hoy en día donde las personas están con menos tiempo y paciencia para realizar trámites o asistir a reuniones. La idea se nos ocurrió como consecuencia de haber escuchado varios comentarios negativos sobre el manejo Administrativo en los edificios. Fue por eso que empezamos a recabar información al respecto. Como todos los integrantes del equipo vivimos en edificio, cada uno se encargó de realizar averiguaciones sobre el funcionamiento de los mismos.

Lo que realizamos fue, consultar a algunos residentes sobre qué aspectos les parecían más aburridos o complicados de realizar, en lo que refiere a la operativa diaria: recibo de estados de cuenta, avisos en el ascensor, etc. El resultado fue que los puntos más complicados eran:

- Comunicación en general: horario de entrada/salida de un portero, fecha de pago de gastos comunes, etc.
- Asistencia a reuniones
- Reserva de servicios
- Falta de información sobre tareas de servicio, mantenimiento, etc. por ejemplo en cañerías o fumigación.

Como primer paso para lograr desarrollar una solución que satisfaga las expectativas de los posibles usuarios, volvimos a consultar a los mismos residentes consultados anteriormente buscando relevar las necesidades y restricciones de los usuarios, para luego analizar la información recabada, especificando y validando los requerimientos resultantes.

## 5.2 Actores del sistema

La aplicación debe soportar dos tipos diferentes de usuarios: residentes y administradores.

Por un lado, están los residentes, quienes controlan su actividad en el edificio en lo que refiere a los intereses compartidos con los demás residentes. Es decir, uso/alquiler de espacios comunes, asistencia a reuniones, acceso a información, etc.

Dentro de los residentes, existe un administrador (uno por edificio) quien se encarga de mantener actualizado la mayoría de las cosas que ocurren en el edificio, tales como reuniones o gestión de servicios.

Por otra parte, los administradores son quienes manipulan y se encargan del mantenimiento de los residentes en general.

## 5.3 Alcance del Producto

El sistema permitirá:

- A los usuarios registrados que residan en uno de los edificios registrados y administrados por la aplicación, poder intercambiar información, administrar recursos y tomar decisiones en conjunto.
- A los usuarios administradores se les ofrecerá acceso a una interfaz que les permitirá un conjunto de funcionalidades preferenciales con el fin de administrar los edificios, disponibilizando información para los residentes, agendando reuniones, entre otras.

Creemos que esta aplicación apunta a un gran público ya que es de utilidad para cualquier persona que viva en un edificio, cooperativa o vivienda donde deba administrar recursos e intercambiar información en conjunto con sus vecinos. La misma resuelve problemas comunes a la mayoría de ellos y facilita la gestión del edificio así como la convivencia entre los residentes.

La implementación del sistema como aplicación móvil permite sacar provecho a varios de los atributos que presentan los dispositivos, como la cámara, GPS, acelerómetro, etc. En la especificación de requerimientos detallamos el uso que daremos a cada una de estas, así como la totalidad de funcionalidades

identificadas. El alcance final, es decir, las funcionalidades que realmente serán implementadas se definirán luego de la primer entrega en conjunto con la cátedra.

## 5.4 Especificación de requerimientos

### 5.4.1 Propósito

Este documento de especificación de requerimientos es el resultado del proceso de Ingeniería de Requerimientos mencionado anteriormente, que sirvió como referencia para conocer a los usuarios del sistema. También resultó útil para el equipo, el cual se encargará del desarrollo del sistema, proporcionando información que luego será analizada para determinar cuál será el alcance que demandará el proceso.

### 5.4.2 Especificación de requerimientos

#### 5.4.2.1 Requerimientos funcionales

A continuación se describen los requerimientos funcionales que describen el comportamiento esperado del sistema. En cada uno de ellos se identifican los casos de uso, que se detallan más adelante, asociados.

##### **Defensa del primer obligatorio**

Luego de la reunión con el cliente (primer defensa de obligatorio), resolvimos que era relevante priorizar los requerimientos funcionales de forma de redefinir el alcance del producto. Los valores para llevar esto a cabo son : Alta/Baja, y nos permitirán guiar el desarrollo de los mismos según su importancia.

##### **RF1: Gestión de Usuarios Administradores**

Descripción:

Los administradores tendrán acceso privilegiado a ciertas funcionalidades con el fin de publicar información para los residentes del/los edificios que administran. Los mismos contarán con una interfaz específica.

El sistema se entregará con las credenciales de un usuario administrador. Este usuario podrá gestionar el alta, baja y modificación de otros usuarios administradores.

El registro de un nuevo usuario administrador requerirá la siguiente información del mismo:

- Nombre
- Email
- Contraseña
- Listado de edificios sobre los cuales el usuario tiene permiso de gestión.

Casos de Uso asociados: CU1, CU2

Prioridad: Alta

### **RF2: Gestión de Usuarios Residentes**

Descripción:

Los residentes de los edificios deberán contactar a un usuario administrador del edificio para solicitar acceso a la aplicación. Este proceso se realiza por fuera del alcance de la aplicación.

El usuario administrador registrará al usuario proporcionando la siguiente información sobre el mismo:

- Nombre
- Email
- Teléfono
- Edificio
- Apartamento

Una vez que el usuario es registrado, se generará un código de autenticación que será entregado al mismo y será necesario en el primer acceso a la aplicación.

Casos de Uso asociados: CU3

Prioridad: Alta

### **RF3: Inicio de Sesión - Administradores**

Descripción:

Los usuarios administradores deberán ser capaces de loguearse en el sistema de administración (interfaz web) utilizando su correo electrónico y contraseña. En caso que el mismo olvidé su contraseña, deberá contactar a otro usuario administrador con acceso al sistema que será capaz de modificarla.

Casos de Uso asociados: CU1

Prioridad: Alta

#### **RF4: Inicio de Sesión - Residentes**

Descripción:

Los usuarios residentes deberán proveer el código de autenticación provisto por el administrador en el primer acceso a la aplicación. Luego de este acceso inicial, el usuario podrá acceder a la aplicación desde su dispositivo sin necesidad de autenticarse.

Casos de Uso asociados: CU2

Prioridad: Alta

#### **RF5: Publicación de Anuncios**

Descripción:

Esta funcionalidad es tanto para administradores como residentes. Cada tipo de usuario tendrá acceso a esta funcionalidad en su interfaz correspondiente. Los anuncios pueden incluir fotos tomadas con el celular o ya almacenadas en el dispositivo del usuario.

Casos de Uso asociados: CU3, CU4, CU5.

Prioridad: Alta

#### **RF6: Visualización de Anuncios**

Descripción:

Aquellos mensajes publicados por los administradores disparan una notificación en el celular de los residentes del edificio. Por otra parte, tanto los administradores como residentes tendrán acceso a una sección específica del sistema donde se listarán las publicaciones.

Casos de Uso asociados: CU3, CU4, CU5.

Prioridad: Alta

#### **RF7: Baja de Anuncios**

Descripción:

Los usuarios administradores del edificio tendrán además la posibilidad de eliminar aquellos anuncios que consideren inapropiados.

Casos de Uso asociados: CU5.

Prioridad: Baja



**RF8: Anuncios Favoritos**

Descripción:

Los residentes podrán seleccionar aquellas publicaciones en las que tienen especial interés como favoritas. Esta categorización les permitirá filtrar los anuncios y obtener rápido acceso a sus anuncios favoritos.

Casos de Uso asociados: CU6

Prioridad: Alta

**RF9: Compartir Anuncios**

Descripción:

Los residentes podrán compartir en Facebook los anuncios a los que tengas acceso.

Casos de Uso asociados: CU17

Prioridad: Alta

**RF10: Publicación de Estados de Cuenta**

Descripción:

Los administradores podrán publicar los estados de cuenta de cada mes con información sobre los gastos realizados en el mes y la deuda de gastos comunes correspondiente a cada apartamento/casa.

Esta información estará disponible para todos los residentes del edificio.

Casos de Uso asociados: CU7

Prioridad: Baja

**RF11: Visualización de Estados de Cuenta**

Descripción:

Los residentes tendrán acceso al archivo histórico de Estados de Cuenta publicados por los administradores del edificio.

Casos de Uso asociados: CU7

Prioridad: Alta

## **RF12: Coordinación de Reuniones de Residentes**

Descripción:

La aplicación permitirá al administrador la coordinación de reuniones de residentes. La coordinación de una reunión generará la creación automática de un anuncio, el cual estará disponible en la visualización de anuncios y por esta ser ingresada por el administrador, generará también una notificación en el dispositivos de los residentes.

Adicionalmente, se creará un evento en el calendario del dispositivo del residente.

Casos de Uso asociados: CU9.

Prioridad: Alta

## **RF13: Recordatorio de Reuniones de Residentes**

Descripción:

La aplicación utilizará el GPS del dispositivo para identificar la ubicación actual del residente. Durante el día en el cual existe una reunión de residentes agendada, el usuario recibirá una notificación en el dispositivo cuando se encuentre dentro de un radio de 500 metros de la ubicación de la reunión.

Casos de Uso asociados: CU8.

Prioridad: Alta

## **RF14: Creación de Votaciones**

Descripción:

Las votaciones serán creadas por el usuario administrador. El mismo podrá exponer a la votación de los residentes aquellos asuntos que surjan durante las reuniones y que deberán resolverse con la opinión de todos los residentes.

El cierre de las votaciones se establecerá durante la creación del mismo.

Casos de Uso asociados: CU9

Prioridad: Alta

## **RF15: Votación**

Descripción:

Los residentes tendrán acceso a las votaciones creadas por el administrador y podrán presentar su voto, el cual una vez presentado no podrá ser modificado.

Casos de Uso asociados: CU10

Prioridad: Alta

#### **RF16: Gestión de Salones o Servicios Comunes**

Descripción:

Los administradores del edificio deberán dar de alta accesos servicios o salones compartidos entre los residentes.

Casos de Uso asociados: N/A

Prioridad: Alta

#### **RF17: Reserva de Salones o Servicios Comunes**

Descripción:

Los residentes podrán reservar la utilización de los servicios o salones comunes que estén disponibles. Al solicitar una fecha y hora, la aplicación informará si dicho horario ya se encuentra reservado y en otro caso realizará la reserva.

Casos de Uso asociados: CU13, CU14

Prioridad: Alta

#### **RF18: Registro de Contactos**

Descripción:

Durante el registro de un nuevo usuario, la creación del mismo como contacto se realizará en forma automática. Adicionalmente, los administradores podrán registrar a otros residentes como contactos.

Casos de Uso asociados: CU15, CU16

Prioridad: Alta

#### **RF19: Agenda de Contactos**

Descripción:

Los residentes tendrán acceso a una agenda donde se visualizan la información de los demás residentes del edificio.

La aplicación permitirá a los usuarios guardar aquellos contactos que sean de su interés en la agenda de contactos de su dispositivo.

Casos de Uso asociados: CU15, CU16, CU17

Prioridad: Baja

### **5.4.2.2 Requerimientos No Funcionales**

#### **RNF1**

Descripción:

El sistema debe estar disponible como una aplicación para dispositivos con Sistema Operativo Android.

#### **RNF2**

Descripción:

La aplicación para Android deberá construirse utilizando Android Studio.

#### **RNF3**

Descripción:

La Web API deberá implementando utilizando el entorno de desarrollo integrado (IDE) Visual Studio y el lenguaje de programación C#.

#### **RNF4**

Descripción:

Para la persistencia de datos deberá utilizarse el Motor de Base de Datos SQL

## **5.5 Cambios en el Alcance**

Durante la semana del 4 de Junio, luego de dedicar X horas al requerimiento RF13 decidimos solicitar a la cátedra un cambio en el scope por los siguientes motivos:

- Conflictos entre las librerías de android 4 para Google Services de nuestro teléfono de prueba y geofences que no pudimos resolver.
- Si continuamos invirtiendo tiempo en tratar de resolver los conflictos mencionados, arriesgábamos no poder resolverlo y terminar con una entrega con alcance menor al establecido.
- Consideramos que el requerimiento no aportaba tanto valor a los usuarios ya que al registrarse las Reuniones de Residentes en el Calendario del dispositivo, los usuarios recibirán una notificación nativa del teléfono de todas formas.

Este riesgo fue identificado durante la etapa de análisis y la mitigación asociada constaba de dedicar mayor tiempo al requerimiento para poder completar el requerimiento en tiempo y forma. En primer aplicamos este enfoque, y luego recurrimos al juicio experto. Sin embargo luego de un par de semanas fue inminente que no íbamos a poder resolver este problema a tiempo de la entrega.

## **5.6 Casos de uso**

Decidimos crear los casos de uso de forma de mostrar las interacciones que desarrollan los actores y nuestro sistema en respuesta a un evento. Esto nos sirvió para especificar la comunicación y el comportamiento del sistema mediante su

interacción con los actores involucrados.

<b>CU1:</b> Inicio de sesión administrador	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se completan los campos de usuario y contraseña correctamente.</li> <li>2. Se presiona el botón de ingreso.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 Se completan los campos de usuario y contraseña correctamente, pero el nombre de usuario ingresado no existe en el sistema.	
1.2 Se completan los campos de usuario y contraseña correctamente, pero la password del usuario ingresado es incorrecta.	
1.3 Se completa el campo de usuario, pero el campo de password se deja vacío, y es Required.	
1.4 Se completa el campo de password, pero el campo de usuario se deja vacío, y es Required.	
1.5 Se dejan en blanco el nombre de usuario y la contraseña, los cuales son Required.	
Requerimientos asociados:	RF3

<b>CU2:</b> Inicio de sesión residente	
<b>Actor:</b> Residente	
<b>Precondición:</b> El residente recibirá un código para activar su usuario en la aplicación por única vez, el cual es solicitado al administrador.	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. El residente recibe código de activación para loguearse al sistema.</li> <li>2. Ingresa el código recibido por única vez.</li> <li>3. Se presiona el botón de ingreso</li> <li>4. El sistema acepta el código ingresado y el residente queda logueado al sistema.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 El residente no recibe código de activación.	
2.1 El residente ingresa un código de activación incorrecto, diferente al que le mandó el administrador.	
2.2 El residente no ingresa ningún código de activación, el cual es Required para loguearse.	

4.1 El sistema no acepta el código y el residente no puede ingresar al sistema.	
Requerimientos asociados:	RF4

<b>CU3: Alta de Usuario</b>	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
5. El administrador completado lo datos del usuario a registrar. 6. Se presiona el botón de ingreso 7. El sistema acepta el la información ingresada y se registra el usuario correctamente.	
<b>Cursos alternativos:</b>	
1.1 El Administrador no completa alguno de los camps requeridos.	
2.1 El usuario que se intenta crear ya existe.	
Requerimientos asociados:	RF1, RF2

<b>CU4: Alta de anuncio</b>	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
Administrador: <ol style="list-style-type: none"> <li>Se completa el campo de texto y/o se adjunta/n alguna/s foto/s.</li> <li>Se presiona el botón para publicar el anuncio.</li> </ol> Residente <ol style="list-style-type: none"> <li>Recibe notificación avisando de la publicación de un nuevo anuncio.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 Se deja vacío el campo de texto y no se adjunta ninguna imagen. 2.1 Si se dejó vacío el campo de texto y no se adjuntó imagen, da mensaje de error.	
Requerimientos asociados:	RF5

<b>CU5:</b> Baja de anuncios	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el anuncio a dar de baja.</li> <li>2. Se confirma la baja del anuncio.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún anuncio. 2.1 Si no se seleccionó ningún anuncio, da un mensaje de error.	
Requerimientos asociados:	RF7

<b>CU6:</b> Seleccionar un Anuncio como Favorito	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<b>Cursos alternativos:</b>	
1.1 No existen anuncios para seleccionar	
Requerimientos asociados:	RF8

<b>CU7:</b> Alta de estado de cuenta	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
<b>Administrador:</b> <ol style="list-style-type: none"> <li>1. Se selecciona la opción de subir estado de cuenta.</li> <li>2. Se selecciona el estado de cuenta a subir.</li> <li>3. Se presiona el botón de confirmación para publicar el estado de cuenta.</li> </ol>	
<b>Cursos alternativos:</b>	
2.1 No se selecciona ningún estado de cuenta. 3.1 Si no se seleccionó ningún estado de cuenta, da un mensaje de error.	
Requerimientos asociados:	RF10

<b>CU8:</b> Control de asistencia en la reunión	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
Administrador <ol style="list-style-type: none"> <li>1. A la hora del comienzo de la reunión se localiza a los clientes, contando cuántos de ellos están a menos de 30 metros del punto de reunión</li> <li>2. Se localiza a los residentes invitados.</li> <li>3. Si más del 50% de invitados está a menos de 30 metros de la zona de reunión, se da por comenzada la reunión.</li> </ol> Residente <ol style="list-style-type: none"> <li>1. Los residentes tienen habilitada la ubicación mediante el GPS.</li> </ol>	
<b>Cursos alternativos:</b>	
3.1 Si menos del 50% de los invitados está en esa zona, se espera hasta llegar a la cuota mínima de participantes.	
Requerimientos asociados:	RF13

<b>CU9:</b> Alta de tema de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona la opción de crear tema de votación.</li> <li>2. Se escribe el nuevo tema.</li> <li>3. Se escriben las opciones de voto para los residentes.</li> <li>4. Se confirma el nuevo tema de votación.</li> </ol>	
<b>Cursos alternativos:</b>	
2.1 Se deja vacío el nombre/objetivo del tema de votación.	
3.1 Se escribe una única opción de voto para los residentes, da un mensaje de error	
3.2 No se escriben opciones posibles de voto para los residentes, da un mensaje de error	
Requerimientos asociados:	RF14



<b>CU10:</b> Alta voto	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el tema por el cual votar.</li> <li>2. Se selecciona la opción de votación.</li> <li>3. Se confirma la votación.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún tema de votación.	
1.2 No hay temas de votación disponibles.	
2.1 No se selecciona ninguna opción de votación.	
3.1 Si no se seleccionó ningún tema de votación, da un mensaje de error	
Requerimientos asociados:	RF15

<b>CU11:</b> Consultar voto	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el tema por el cual se votó.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún tema de votación.	
Requerimientos asociados:	RF14, RF15

<b>CU12:</b> Cierre de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el tema por el cual se votó.</li> <li>2. Cuando el administrador lo considere, se selecciona la opción de cerrar la votación.</li> <li>3. Se envía una notificación con el resultado de la votación a todos los residentes.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún tema de votación.	
2.1 No hay temas de votación disponibles, da un mensaje de error.	

2.2 Si votó menos del 50% de los residentes, da un mensaje de error indicando que deben votar más personas.	
Requerimientos asociados:	RF14, RF15

<b>CU13:</b> Alta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el servicio a contratar.</li> <li>2. Se selecciona la fecha/hora de reserva.</li> <li>3. Se confirma la reserva.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún servicio.	
2.1 No se selecciona ninguna fecha.	
3.1 Si se seleccionó una fecha que estaba ocupada, da mensaje de error.	
3.2 Si se seleccionó una fecha anterior al día actual, da mensaje de error.	
Requerimientos asociados:	RF17

<b>CU14:</b> Consulta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona el servicio a consultar.</li> </ol>	
<b>Cursos alternativos:</b>	
1.1 No se selecciona ningún servicio.	
Requerimientos asociados:	RF17

<b>CU15:</b> Alta de contacto en agenda	
<b>Actor:</b> Administrador	
<b>Precondición:</b> Puede existir más de un contacto registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona la opción de alta de contacto en agenda.</li> <li>2. Se escriben los datos del usuario, lo cual es mandatorio.</li> <li>3. Se confirma la creación de usuario.</li> </ol>	
<b>Cursos alternativos:</b>	
2.1 Se deja algún dato del usuario en blanco.	
3.1 Si falta ingresar algún dato del usuario, da un mensaje de error.	
Requerimientos asociados:	RF18

<b>CU16:</b> Consulta de contacto en agenda	
<b>Actor:</b> Administrador – Residente	
<b>Precondición:</b> Puede existir más de un contacto registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. Se selecciona la opción de consultar de usuario en agenda.</li> <li>2. Se selecciona el usuario a consultar.</li> </ol>	
<b>Cursos alternativos:</b>	
2.1 No se selecciona ningún usuario a consultar.	
Requerimientos asociados:	RF19

<b>CU17:</b> Compartir anuncio en Facebook	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
<ol style="list-style-type: none"> <li>1. El residente se loguea a Facebook.</li> <li>2. Comparte el anuncio en facebook.</li> </ol>	
<b>Cursos alternativos:</b>	
1. El usuario no tiene cuenta en Facebook.	
Requerimientos asociados:	RF9

<b>CU18:</b> Registrar Contacto en la Agenda del Celular	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	
3. El residente accede a la lista de Agenda de Contactos de la aplicación. 4. Selecciona un contacto y se abre la agenda del celular para guardarlo	
<b>Cursos alternativos:</b>	
2. No existen contactos en la agenda de la aplicación.	
Requerimientos asociados:	RF18

## 6. Gestión de la comunicación

Para gestionar la comunicación es imprescindible en primera instancia tener claro qué queremos conseguir, cuáles son nuestros objetivos y qué medios utilizaremos para ello. Es por esto que establecimos los canales o medios de comunicación en los que se trabajará a lo largo del proyecto:

Canal de comunicación	Objetivos principales	Frecuencia
Google Drive: Planilla de Registro de esfuerzos	Registro de información	1 o 2 veces por semana
Google Drive: Documentación	Documentación	2 o 3 veces por semana
Trello	Registro de información	1 vez por semana
Repositorio	Actualización de software	1 o 2 veces por semana
Grupo de WhatsApp	Aclaraciones, consultas	4 veces por semana
Skype	Aclaraciones, consultas, reuniones	1, 2 veces por semana
Reuniones periódicas	Aclaraciones, consultas, puesta a punto dentro del equipo, registro de información, actualización del software.	1 vez por semana
Clase	Aprobación de propuesta, consulta de dudas y puesta a punto con el docente.	1 vez por semana

Luego de la reunión con el cliente (defensa del primer obligatorio), decidimos agregar la columna "Frecuencia" que determina cada cuanto tiempo se llevan a cabo estas comunicaciones. Dependiendo del rol de cada uno es cómo se distribuyen las horas mencionadas.

## 6.1 Plan de comunicación

El Plan de comunicaciones del proyecto, es “un componente que describe la forma en que se van a planificar, estructurar, monitorear y controlar las comunicaciones del proyecto”. Así se plantea la Guía de fundamentos para la dirección de proyectos en su quinta edición (PMBOK 5), estándar de Gerencia de proyectos establecido por el Project Management Institute (PMI).

Además, presentaremos una plantilla de Plan de Gestión de comunicaciones del proyecto, que permite registrar: las restricciones y premisas de comunicación, los requisitos de comunicación de los interesados (stakeholders), las comunicaciones a emitir en el proyecto, su objetivo, frecuencia, contenido y responsables de su emisión, los recursos asignados a las comunicaciones, glosario de términos, entre otros.

La plantilla mencionada contiene los siguientes puntos:

1. Información del proyecto: Especifica al equipo de facultad que ejecuta el proyecto, el título del proyecto, el encargado de Gestión de Proyectos asignado, el encargado de Desarrollo, el encargado de Calidad, y la fecha en que se preparó el documento.
2. Recursos asignados: Aquí se especifican tiempos en el registro de esfuerzos, asignados a comunicaciones del proyecto.
3. Actualización y mejoras en el plan: En la medida en que el proyecto se desarrolla, se pueden identificar oportunidades de mejora. En esta sección se especifican los procedimientos para identificar, evaluar y aprobar dichos cambios.
4. Anexos: El Plan de Gestión de Comunicaciones también puede incluir guías, bocetos de diseño y planillas para reuniones del equipo, reuniones electrónicas, formatos para comunicaciones vía correo electrónico, entre otros. Estos se incluyen en la sección de Anexos.

En nuestro caso, de forma de verificar el cumplimiento de todos estos puntos en la presente documentación, hicimos la siguiente tabla. La misma detalla la realización de los distintos puntos de comunicación que detallaba la plantilla anterior:

<b>Nro.</b>	<b>Nombre</b>	<b>Implementación (en donde se realizó)</b>
1	Información del proyecto	1.Objetivos. 2.Tecnologías y herramientas desarrolladas
2	Recursos asignados a actividades	4. Asignación de roles.
3	Actualización y mejoras en el plan	8. Gestión de la configuración.
4	Anexos	14. Anexos

## 7. Gestión de proyecto

### 7.1 Cronograma

Como mencionamos en la sección “Metodologías Utilizadas” seguiremos las fases establecidas en el ciclo de vida cascada. Identificamos como hitos principales del proyecto las dos entregas del obligatorio. A continuación describimos el alcance y los objetivos para cada una de ellas:

#### **Primera entrega: 27/04/2017**

Para ésta primera instancia, se presenta el detalle del alcance y proceso que involucra el desarrollo de la aplicación. En particular mostraremos un análisis detallado de la solución, justificando cada decisión tomada desde el tipo de enfoque metodológico elegido, hasta la aplicación de los elementos que nos brinda el dispositivo: GPS, cámara, etc.

Esta primera instancia abarca principalmente las fases de Análisis y Diseño. En paralelo a estas actividades, se llevarán a cabo pruebas de concepto y tareas de investigación con el fin de familiarizarnos con la tecnología a utilizar durante la segunda instancia.

#### **Reunión con el cliente (Defensa primer entrega): 8/05/2017**

Durante esta instancia, se refinó con el cliente los requerimientos que se implementarán posteriormente en la etapa de desarrollo. Ésto implicó una revisión y devolución por parte del mismo, en donde nos indicó cuáles son las prioridades de cada uno de los requerimientos. Se definió el alcance final que tendrá el producto, así como varias oportunidades de mejora en lo que refiere al proceso, proyecto y producto.

#### **Segunda entrega: 22/06/2017**

En la próxima instancia, se presentará el producto terminado, lo que incluye: la finalización del proceso de desarrollo de software, abarcando documentación detallada del control de calidad, diseño, arquitectura, implementación, y demo a entregar al cliente. Esta instancia abarca las siguientes fases: continuación del Diseño (a más bajo nivel), Implementación y Pruebas.

#### **Demo al Cliente: fecha a definir**

En esta instancia se realizará una demo donde se presentará el producto



finalizado al cliente. Se probarán las diferentes funcionalidades implementadas.

## 7.2 Gestión de tareas

La gestión de tareas será principalmente coordinada y supervisada por el Responsable de la Gestión de Proyecto.

Con el fin de obtener un repositorio colaborativo, donde cada miembro del equipo pueda estar al tanto e informar al equipo de su propio avance, utilizaremos la herramienta Trello. Esta es una herramienta para la gestión de tareas colaborativa. Crearemos un tablero específico para el proyecto donde utilizaremos cuatro listas:

- To Do: Tareas pendientes a llevar a cabo
- Doing: Tareas en proceso de realización
- Done: Tareas completadas
- Blocked: Tareas que se encuentran bloqueadas por problemas técnicos o de otra índole.

Cada una de las tarjetas, tendrá a el o los miembros del equipo asignados para llevar a cabo la tarea que representa.

A su vez, detalla la estimación realizada para tarea.

El Responsable de Calidad será el encargado de validar que las tareas que se encuentren en la lista “Done” cumplir con los requisitos necesarios para poder considerarse como finalizadas. Utilizaremos la funcionalidad de “Checklists” con el fin de reportar errores o tareas adicionales dentro de cada tarjeta. De esta forma, a medida que los errores se corrijan o las tareas se realicen, el miembro que lo completa puede seleccionar el ítem de la lista como finalizado.

El acceso al tablero el Trello es público y se puede acceder a través del siguiente enlace: <https://trello.com/b/388S9ern/vecinosuy>

## 7.3 Gestión de riesgos

### 7.3.1 Identificación de Riesgos

A continuación detallamos los riesgos identificados que pueden tener un impacto en el transcurso del proyecto y su categorización. Para la categorización consideramos dos tipos de riesgos: Tecnológicos y de Proyecto. Los riesgos tecnológicos afectan a la planificación, el costo y calidad del proyecto. Identifican

problemas potenciales de, calendario, personales, recursos, etc.

Los riesgos tecnológicos amenazan la calidad del software (producto) a producir. Identifican posibles problemas de incertidumbre técnica, ambigüedad en la especificación, diseño, implementación, etc.

Tipo de Riesgo	Descripción
Tecnológico	Desconocimiento de la tecnología a utilizar para la implementación de la aplicación para Android. El desconocimiento no aplica específicamente para el sistema operativo sino sobre el desarrollo de aplicación para mobile en general.
De Proyecto	Errores en las estimaciones debido a la falta de experiencia del equipo.
De Proyecto	Mala utilización del tiempo para invertir en el proyecto puede generar que no se cumplan las tareas requeridas para la fecha estipulada.
De Proyecto	Diseño inadecuado del software (implica un re-diseño) debido a la falta de experiencia del equipo.
De Proyecto	Se pueden generar conflictos internos entre los miembros del equipo de proyecto si no saben manejar de forma correcta los problemas que surjan tanto en la interna como en el desarrollo del producto.

### 7.3.2 Análisis de Riesgos

A continuación detallamos el análisis de los riesgos identificados en la sección anterior. Para el análisis cualitativo realizamos una evaluación de tipo de riesgos, elaboramos un plan de respuesta de los riesgos, en el cual se mencionan diferentes estrategias: mitigar, eliminar, transferir y aceptar, y por último un plan de contingencia.

Luego de la reunión con el cliente (primer defensa de obligatorio), agregamos más información a la tabla de riesgos. En particular, para medir la prioridad del riesgo, definimos la escala Alto/Bajo, en donde se hizo hincapié en minimizar de la mejor forma posibles problemas. Por otro lado, para medir el impacto de los mismos se definió una escala del 1 al 5, en donde se cuantificó qué tanto impacto puede tener ese riesgo en el proyecto. En la tabla muestra cómo cuantificamos los riesgos al comienzo de la etapa de desarrollo.

Como resultado del análisis, realizamos la siguiente tabla:

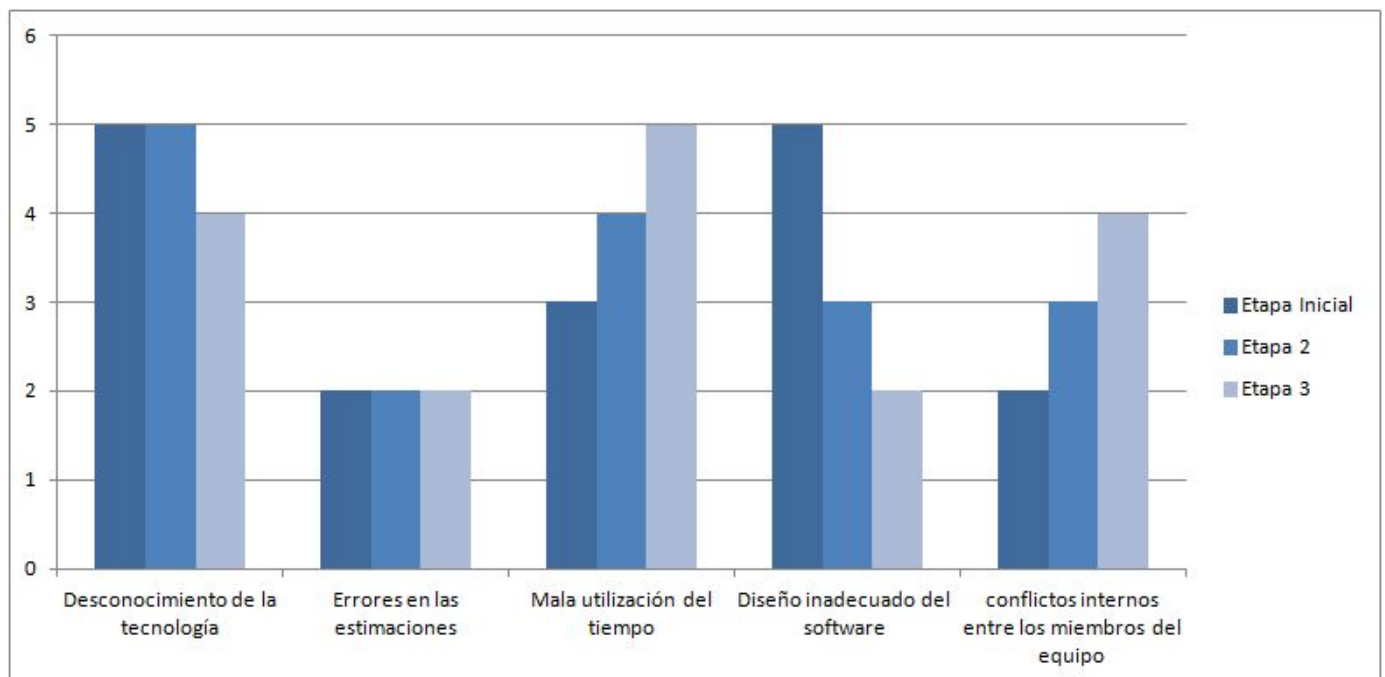
<b>Prioridad Inicial (Alta/Media/Baja)</b>	<b>Riesgo</b>	<b>Plan de Respuesta</b>	<b>Plan de Contingencia</b>	<b>Impacto Inicial (1,2,3,4,5)</b>
Alta	Desconocimiento de la tecnología a utilizar para la implementación de la aplicación para Android. El desconocimiento no aplica específicamente para el sistema operativo sino sobre el desarrollo de aplicación para mobile en general.	Mitigar: Todo el equipo estudiará de forma rigurosa la nueva tecnología. Se buscarán fuentes de información como son las guías de clase, o sitios confiables de internet con manuales/libros, etc.	Se extenderá la duración de las reuniones y la dedicación personal (de cada miembro del equipo) al trabajo de forma de aprender la tecnología sin que esto afecte las fechas y los objetivos pautados.	5
Media	Errores en las estimaciones debido a la falta de experiencia del equipo.	Mitigar: Se realizará la técnica de juicio experto con el fin de obtener herramientas y técnicas que nos ayuden con las estimaciones	Se consultará a expertos en la materia que brinden asesoramiento técnico en las estimaciones.	2
Media	Mala utilización del tiempo para invertir en el proyecto puede generar que no se cumplan las tareas requeridas para la fecha estipulada.	Aceptación activa: se buscará aprovechar de la mejor manera el tiempo invertido en el proyecto.	Se contará con noches sin dormir de forma de reforzar la falta de tiempo sin que afecte la planificación del proyecto.	3
Alta	Diseño inadecuado del software (hay que volver a diseñar) debido a la falta de experiencia del equipo.	Mitigar: Se realizará un análisis detallado de los requerimientos de forma de verificar que los mismos están alineados con diseño planteado por el equipo de proyecto.	Se utilizará la técnica de juicio experto, en donde se consultará a expertos sobre cómo modificar el diseño de la mejor manera.	5
Baja	Se pueden generar conflictos internos entre los miembros del equipo de proyecto si no saben manejar de forma correcta los problemas que surjan tanto en la interna como en el desarrollo del producto.	Mitigar: Generar actividades periódicas que ayuden al equipo a mantenerse organizado, enfocado, relajado y comprometido sin que algún conflicto interno/externo los perjudique.	Se buscará realizar dinámicas motivacionales en grupo, que ayuden a manejar los problemas de forma que no afecte el proyecto.	2

## Evolución de los riesgos en el tiempo

El comportamiento de los riesgos determina aspectos importantes de nuestro proceso. Es por eso que consideramos relevante registrar la evolución de los mismos en el tiempo. Por este motivo, decidimos realizar una evaluación de los mismos cada 14 días. Esto nos permite conocer el comportamiento y evolución de los mismos, permitiéndonos tomar precauciones en caso de obtener resultados muy desfavorables o inferiores a lo

esperado. Se espera que los riesgos más prioritarios y con mayor impacto vayan reduciendo en el correr de los días de la mejor forma posible.

A continuación se muestra una gráfica que muestra este comportamiento en el correr del tiempo:



El resultado de la misma muestra como algunos riesgos progresaron en el tiempo. A continuación se detalla un análisis del comportamiento de cada uno de ellos:

**Desconocimiento de la tecnología:** Este riesgo se mantuvo con un impacto muy alto durante las primeras etapas ya que durante toda la implementación nos encontramos constantemente con desafíos en el desarrollo causados por esta razón. En la última etapa el riesgo bajo ya que la misma no contó con tantos esfuerzos de desarrollo como las anteriores.

**Errores en las estimaciones:** Debido a que las estimaciones se realizan en su totalidad al principio de proyecto, el riesgo de que las mismas no fueran correctas se mantuvo constante durante todo el proyecto.

**Mala utilización de tiempo:** Este riesgo aumentó gradualmente al acercarse la fecha de la entrega final, ya que de materializarse el mismo implicaba una entrega incompleta.

**Diseño inadecuado del software:** Este riesgo fue muy alto durante las primeras etapas de desarrollo pero disminuyó a lo largo del proyecto.

**Conflictos internos entre los miembros del equipo:** Este riesgo se mantuvo bajo ya que los miembros del equipo ya tenían experiencia trabajando en conjunto. Sin embargo, el riesgo del mismo aumentó a medida que la fecha de la entrega se acercaba.

## 8. Gestión de la configuración

### 8.1 Control de cambios y de versiones

Se definirán herramientas para gestionar, controlar y proteger el almacenamiento adecuado del programa y documentación, así como: prevenir el acceso sin autorización, realizar seguimiento del estado de las versiones y sus cambios. Estas herramientas componen lo que llamamos la Gestión de Configuración de Software (SCM).

#### 8.1.1 Solicitudes de cambio

El control de cambios se realiza en los proyectos para evitar el problema que pueden generar los cambios no controlados. Existen distintas herramientas diseñadas para realizar el seguimiento de un proceso, incluidas las solicitudes de cambio. Para este proyecto en particular, utilizaremos Trello, donde cualquier cambio deberá registrarse y ser aprobado antes de llevarse a cabo. Esto nos va a permitir tener controlados los cambios y obtener una visión y control de las tareas en donde todas las personas involucradas tendrán acceso. Además, esta herramienta permite indicar quien tiene asignada una tarea, cuál es la prioridad de la misma, etc.

La Gestión de solicitudes de cambio se dará cuando se encuentren errores a la hora de probar el software. En nuestro caso nosotros vamos a crear las solicitudes de cambio en caso de que sea necesario, por ejemplo al realizar la etapa de pruebas de alguna funcionalidad. Para el control de cambios es importante crear informes de forma que el equipo de proyecto esté al tanto de todos cambios realizados.

Si encontramos bugs podremos crear nuevas tareas que serán asignadas y evaluadas por quien se encargue del desarrollo. Para estos bugs el encargado de programar hará una evaluación, y en función de la decisión que tome, asignará este bug a los programadores. En el caso que se apruebe el cambio, el encargado va a especificar una prioridad a cada defecto. Por ejemplo si el error encontrado es de vida para el software, deberá asignarle prioridad alta de forma que los desarrolladores le den mayor importancia y prioridad a solucionarlo.

Este proceso se va a realizar también cuando implementemos las

modificaciones sobre el producto que nos solicitaron. Al realizar cambios en el código, estos deberían verse reflejados también en el repositorio. Para esto, luego que se solicita el cambio y se modifica el código, se debe hacer el Commit correspondiente. De esta forma, los cambios quedan registrados correctamente.

### 8.1.2 Versionado

Es importante el versionado de los datos a la hora de encontrar errores ya que gracias al mismo se pueden ver las versiones anteriores y encontrar de donde viene el problema para solucionarlo, y además se podría encontrar el responsable del mismo. Y en el caso que sea necesario revertir los cambios que fueron ingresados.

Consideramos relevante que se controlen los conflictos generados con las diferentes versiones de código, así como: evitar que se pierdan los cambios realizados por los programadores o que quede código inconsistente en el repositorio.

En nuestro caso, lo manejaremos de la siguiente forma:

- Para el versionado de la documentación utilizamos Google Drive, en donde cada integrante modificó la misma de manera colaborativa. A través del plan de comunicación y de la facilidad que brinda Google Drive al avisar cuando alguien más está editando un archivo, se intentó tener acceso controlado de forma de no “pisarnos” el trabajo.
- Para el versionado del producto vamos a utilizar el repositorio Git de Bitbucket, que nos provee un repositorio del código donde cada desarrollador actualiza los cambios realizados en el código. Quien realiza la actualización escribe una breve descripción de lo que hizo. De esta forma cada integrante del equipo sabe en qué situación se encuentra el desarrollo, así como tiene conocimiento de la trazabilidad del desarrollo y los cambios que realizó cada integrante, lo que evita el re-trabajo.

### 8.1.3 Gestión de defectos

Un defecto es la imperfección que causa el fallo, la cual puede ser visible o no en el exterior del producto. Es decir, puede tratarse de una anomalía interna del producto. Un defecto puede existir en el producto y no manifestarse como fallo. Por ejemplo, es un defecto tener una variable no inicializada, lo cual puede causar un fallo.

La gestión de defectos es una parte implícita de la fase de ejecución, pero que al tener una alta importancia en las pruebas funcionales, diferenciamos como una etapa independiente. Cuando al realizar un paso el resultado obtenido no es el esperado, habrá que reportar un defecto para que el equipo de desarrollo tenga conocimiento del error.

Una técnica muy efectiva para detectar defectos en el producto son las revisiones. Éstas son económicas y permiten detectar defectos antes que las pruebas. En este caso, se implementará el tipo de revisión denominada: “Revisión de a pares”, la cual consiste en que un integrante del equipo realiza una copia del artefacto a probar, lo revisa y retorna la lista de defectos detectados. En nuestro caso es una técnica muy conveniente ya que contamos con poco tiempo y escasos recursos.

Los defectos deben ser claros y detallados, tienen que describir el error para que el mismo sea comprenderlo perfectamente, reproducirlo, localizarlo y poder solucionarlo. Se deberá realizar un monitoreo continuo del estado de los defectos y de esta forma poder realizar las pruebas necesarias para su finalización.

Formato para el reporte de defectos:

Identificación del defecto	
Descripción	
Gravedad (Baja/Media/Alta)	
Rol responsable	
Nombre responsable	
Solucionado (S/N)	

## 9. Aseguramiento de la calidad

### 9.1 Proceso y producto

Construiremos un plan de aseguramiento de la calidad de forma de especificar las actividades que se van a realizar para asegurar la calidad del software a construir. En él se detalla que es lo que se va a realizar: revisión del uso apropiado de estándares, normas, métodos, procedimientos, etc. Existen diferentes actividades que nos permiten asegurar la calidad de un producto: preventivas, correctivas y registrales. Las mismas requieren de la cooperación de equipo de desarrollo para ser llevadas a cabo de forma correcta por el encargado de SQA . Una tarea se considerará completa si se generó un reporte acerca de la misma.

#### 9.1.1 Análisis de calidad

Los objetivos de quien se encarga de las actividades de SQA consiste en obtener y mantener la calidad de: requerimientos, diseño, código y la efectividad del control de calidad.

El análisis de calidad de los requerimientos establece un único entendimiento del problema entre el cliente (el profesor) y el equipo de desarrollo, y consta de las siguientes actividades:

- Revisar todos los requerimientos para determinar si su implementación es factible.
- Verificar y analizar los requisitos que puedan tener algún tipo de error. En el caso que los mismos tengan que ser modificados, darle un seguimiento a los cambios.

En nuestro caso, como tomamos al profesor como cliente, utilizamos las instancias de clase para plantear dudas o sugerencias sobre nuestro proyecto. Durante estas instancias conversamos sobre las posibles mejoras o modificaciones que podíamos realizarle al producto.

Las tareas relacionadas a la calidad de diseño son:

- Verificar que el proceso de diseño de software siga el estándar determinado.
- Verificar que todos los elementos que no cumplen con la calidad requerida sean procesados de acuerdo a lo establecido.
- Verificar que el diseño corresponde con los requerimientos previamente determinados.



#### 9.1.1.1 Actividades preventivas

Estas actividades se llevan a cabo con el fin de detectar posibles errores y desviaciones en el proceso. Las mismas tienen de mucha importancia en los planes de Aseguramiento de calidad ya que es muy importante poder encontrar las fallas antes que se propaguen. Cuanto antes se encuentren los problemas en el software, se necesita menos tiempo y dinero para solucionarlas. Las actividades preventivas se llevan a cabo mediante revisiones, en las cuales se pueden detectar distintos tipos de errores, y luego corregirlos.

A continuación explicaremos algunas actividades de evaluación que se llevarán a cabo de forma de poder detectar problemas y corregirlos:

- Enfocarnos en las pruebas para que los factores de riesgo se detecten mediante el monitoreo, para que las conexiones con sistemas de emergencia no fallen, etc.
- Verificar que no haya cambios no documentados.
- Realizar análisis causal de forma de conocer la causa de los errores de forma de poder solucionarlos correctamente.

En este caso, se realizará un análisis de forma semanal por parte del encargado de Calidad al software, de forma de detectar problemas e identificar tendencias que puedan causar problemas genéricos en las áreas. Destacamos la importancia de anticiparnos a los problemas y no dejarlos para más adelante. Debemos evitar la “Deuda técnica”, una metáfora de las consecuencias de tomar decisiones que comprometan el diseño y aumenten costos futuros de mantenimiento.

Las tareas preventivas a realizar son:

- Probar/mostrar funcionalidades a individuos ajenos al desarrollo del proyecto (conocidos, amigos, compañeros de trabajo, etc.)
- Mostrar el diseño de UI del producto mediante mockups con individuos mayores de edad, que eventualmente se encuentren más alejados de las tecnologías (vecinos, familiares, etc.).

#### 9.1.1.2 Actividades correctivas

Este tipo de actividades se realizan con el fin de realizar correcciones en base a problemas detectados, para lo cual se definen acciones correctivas y de esta manera se mejora la productividad del equipo. En este caso quien se encarga de la Calidad quien debe analizar los problemas detectados, su causa, impacto y frecuencia. Por ende, lo que se realiza es la corrección de defectos en la etapa de testeo.

Luego de que cada versión del software esté disponible, el encargado de Calidad se encargó de detectar posibles correcciones que ayuden a mejorar el trabajo. Las correcciones más frecuentes fueron las siguientes:

- Algunas diferencias de interpretación entre los miembros del equipo con respecto a los requerimientos especificados, cuya acción correctiva radicó en cambios en el producto.
- Detalles de usabilidad, que se modificaron a medida que iban surgiendo nuevas versiones del producto.
- Arreglar bugs/errores en funcionalidades.
- Corregir casos denominados “borde”.

Es importante aclarar que estos tipos de correcciones son bastante discutibles y fueron charladas y justificadas en instancias donde cada miembro del equipo dió su opinión, buscando llegar a un “acuerdo”.

Las tareas correctivas se realizarán, en caso de ser necesario, como consecuencia de las siguientes actividades:

- Ejecución de pruebas unitarias
- Ejecución de pruebas de integración
- Ejecución de pruebas de aceptación
- Prueba del software por parte de usuarios ajenos al equipo de proyecto.

#### 9.1.1.3 Actividades registrales

El objetivo de estas actividades es documentar todas las tareas que se realizaron y sus resultados obtenidos, de manera que quede registrado y todo el equipo pueda estar informado sobre los inconvenientes que puedan surgir. Esto sirve de guía para determinar qué aspectos se deberían mejorar, ya sea desde el punto de vista de las funcionalidades del producto, o cualquier otro aspecto relacionado al mismo.

#### 9.1.1.3.1 Registro de Esfuerzos

En particular para este proyecto, realizamos:

- Un registro de horas de todo el equipo en donde cada miembro especificó el tiempo

que le dedicó a cada tarea, la fecha y detalles de lo que estaba trabajando en el momento.

- Como mencionamos en la parte de 6.1.2 Versionado, cada vez que se actualizó el software, se commiteó (en Bitbucket) escribiendo una descripción detallada del trabajo realizado. Ésto nos aporta una trazabilidad del proceso de desarrollo.

El registro de horas se realizó a través de una planilla compartida a través de Google Docs (de Google Drive), en donde cada integrante fue completándola de manera colaborativa.

En los Anexos parte 9.1.1.3 Actividades registrales se adjunta el registro de horas.

#### 9.1.1.3.2 Estimaciones

Se realizaron estimaciones a lo que refiere al desarrollo de los requerimientos propuestos. Este trabajo fue realizado por el Encargado de Desarrollo, quien cuenta con 5 años de experiencia en el área de desarrollo de software. Estas estimaciones se realizaron en forma empírica, en base a la experiencia pasadas y conocimiento del trabajo a realizar.

En los Anexos parte 9.1.1.3 Actividades registrales se adjunta el detalle de las estimaciones de los requerimientos.

### 9.1.2 Plan de métricas

Para definir un plan de métricas se deberían utilizar métricas de producto, proceso y proyecto.

Las métricas de producto son útiles para mejorar el producto y nos aportan información sobre medidas acerca del cumplimiento de los requisitos para el producto. Paralelamente, las métricas del proceso nos aportan medidas para poder realizar un seguimiento del proceso y de esta forma poder identificar las oportunidades de mejora.

Creemos muy relevante el hecho de realizar y/o analizar un reporte de métricas para el proyecto ya que, a partir de éstas, se pueden tomar decisiones que permitan mejorar el diseño o estimular futuras fases de desarrollo.

Más allá de las ventajas que esto nos aporta, no deberían ser las métricas la única forma de evaluar la calidad de un software.



En nuestro caso, para obtener el informe de métricas utilizamos la herramienta de Visual Studio denominada Code Metrics. La misma realiza un análisis del código y brinda los siguientes datos para cada uno de los proyectos de la solución:

- **Índice de Mantenibilidad:** Calcula un valor de índice entre 0 y 100 que representa la relativa facilidad de mantenimiento del código. Un valor alto significa una mejor mantenibilidad.
- **Complejidad Ciclomática:** Mide la complejidad estructural del código. Se crea calculando el número de diferentes rutas de código en el flujo del programa
- **Profundidad de herencia:** Indica el número de definiciones de clase que se extienden a la raíz de la jerarquía de clases. Cuanto más profunda sea la jerarquía, más difícil será comprender dónde se definen y / o redefinen métodos y campos particulares.
- **Acoplamiento de clases:** Mide el acoplamiento a clases únicas a través de parámetros, variables locales, tipos de retorno, llamadas a métodos, instanciaciones genéricas o de plantilla, clases base, implementaciones de interfaces, campos definidos en tipos externos y decoración de atributos. Un buen diseño de software dicta que los tipos y métodos deben tener alta cohesión y bajo acoplamiento. Alto acoplamiento indica un diseño que es difícil de reutilizar y mantener debido a sus muchas interdependencias en otros tipos.
- **Líneas de código:** indica el número aproximado de líneas en el código. Un recuento muy alto podría indicar que un tipo o método está tratando de hacer demasiado trabajo y debe dividirse. También podría indicar que el tipo o método podría ser difícil de mantener.


A continuación se detalla el análisis obtenido:

Code Metrics Viewer

Analyze Solution

  Compare...

Maintainability Index ▾ Min:  Max:

 Goto Next ▾

Hierarchy	Maintainability Index	Cyclomatic Compl...	Class Coupling	Depth of Inheritance	Lines of Code
VecinosUY.Data.DataAccess.dll	<div></div> 92	22	17	2	23
VecinosUY.Data.dll	<div></div> 93	125	6	1	126
VecinosUY.Exceptions.dll	<div></div> 94	9	2	2	12
VecinosUY.Factory.dll	<div></div> 78	25	14	1	49
VecinosUY.Logger.dll	<div></div> 100	2	2	0	0
VecinosUY.Logic.Test.dll	<div></div> 80	71	55	1	179
VecinosUY.Logic.dll	<div></div> 88	237	46	1	397
VecinosUY.PlainTextLogger.dll	<div></div> 61	11	11	1	36
VecinosUY.Repository.dll	<div></div> 89	65	35	1	92
VecinosUY.Resolver.dll	<div></div> 91	24	25	1	30
VecinosUY.Security.dll	<div></div> 77	9	10	1	18
VecinosUY.Test.dll	<div></div> 81	60	63	1	727
VecinosUY.Web.Api.dll	<div></div> 75	321	89	2	730

Podemos ver que a nivel general el índice de mantenibilidad es alto, superando siempre el 75%.

Por otra parte, la complejidad ciclomática es muy variada en los diferentes proyectos, probablemente debido al tamaño de los mismo.

El acoplamiento de clases también es muy variado. El análisis indica que las proyectos “Web.Api” y “Logic” tienen acoplamiento muy alto y por lo tanto serian muy difícil de reutilizar.

### 9.1.3 Oportunidades de mejora

#### En el proceso:

La principal oportunidad de mejora que identificamos a nivel del proceso, fue optar por una metodología en la que el testing se realice luego de finalizada cada funcionalidad y no luego que todas ellas fueron implementadas. Por seguir un ciclo de vida de cascada, la identificación de bugs se realizó muy cercano a la fecha de entrega y sin tener suficiente tiempo para poder corregirlos.

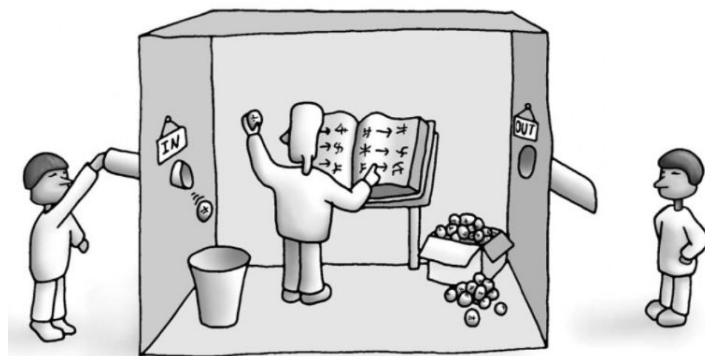
### En el producto:

Mejoras identificadas a nivel de UX y UI:

- No tener que volver para atrás para ver los cambios actualizados, ejemplo durante la creación de un anuncio.
- Mostrar en la página inicial los Anuncios y disponibilizar el resto de las opciones utilizando un menú “hamburguesa”.
- No tener que seleccionar un botón para acceder al Spinner de Fecha/Hora. Seleccionar el datos ahí mismo y poder ver la selección.

## 9.2 Pruebas de software

En esta unidad explicaremos el proceso de llevar a cabo las pruebas en el software desarrollado. Esto incluye desde el diseño, ejecución, resultados y posterior análisis de las mismas.



### 9.3.1 Pruebas unitarias

De forma de probar nuestro código, realizamos pruebas unitarias automatizadas mediante el framework xUnit de test unitarios. El método utilizado es el de caja blanca, en donde elegimos distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y verificar que los mismos devuelven los valores de salida adecuados.

Aplicamos las reglas FIRST sobre el código de cada prueba realizada, las cuales consisten en:

- Se ejecutan de forma rápida.
- Son independientes entre sí.
- Se ejecutan en repetidas ocasiones obteniendo los mismos resultados siempre.
- El propio test compara el resultado obtenido con el correcto y determinar si la prueba funciona correctamente o no.
- Escribimos las pruebas en el momento adecuado, que es antes de

escribir el código, lo que permite escribir luego un código más fácil de probar.

Con las pruebas, mejoramos la calidad del software, asegurándonos de que aunque hagamos cambios en nuestro código, estamos cumpliendo con los requisitos esperados. En un futuro podremos modificar, ampliar o eliminar código y las pruebas unitarias seguirán sirviéndonos para realizar las comprobaciones pertinentes.

La cobertura de código marca la cantidad de código de la aplicación que está sometido a una prueba. En este caso, el nivel de cobertura de los test sobre el código entregado fue de **60,46%**. Intentamos abarcar la mayoría de los casos “borde” encontrados en los casos de uso, de forma de evitar posibles inconvenientes a los futuros usuarios del sistema.

No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
1875	39,54 %	2867	60,46 %

### 9.3.2 Pruebas de aceptación

Este tipo de pruebas sirve para verificar si el producto fue desarrollado acorde a las normas y criterios establecidos y cumple con todos los requerimientos especificados por el cliente. Las mismas se llevan a cabo en general por parte del cliente. El objetivo no radica en validar el comportamiento lógico específico de un componente sino un escenario, caso de uso, etc. La metodología que se aplica es la de caja negra, en donde el usuario evalúa el funcionamiento global del sistema y va comparando los resultados con los requerimientos establecidos inicialmente.

En nuestro caso, consideramos que estas pruebas son de las más importantes, debido al involucramiento que tiene el usuario. Las mismas se realizan antes de que el producto es finalmente entregado. Para esto tomaremos como base los casos de uso y requerimientos funcionales definidos para el sistema.

PA1: Inicio de sesión administrador	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se completan los campos de usuario y contraseña correctamente.	[16-06-2017] Se inició la sesión correctamente
<b>Cursos alternativos:</b>	
Se completan los campos de usuario y	[16-06-2017] Se obtuvo el siguiente mensaje

contraseña correctamente, pero el nombre de usuario ingresado no existe en el sistema.	de error "El usuario especificado no existe o su contraseña es incorrecta"
Se completan los campos de usuario y contraseña correctamente, pero la password del usuario ingresado es incorrecta.	[16-06-2017] Se obtuvo el siguiente mensaje de error "El usuario especificado no existe o su contraseña es incorrecta"
Se completa el campo de usuario, pero el campo de password se deja vacío, y es Required.	[16-06-2017] Se obtuvo el siguiente mensaje de error "Completa este campo" indicando el campo para usuario
Se completa el campo de password, pero el campo de usuario se deja vacío, y es Required.	[16-06-2017] Se obtuvo el siguiente mensaje de error "Completa este campo" indicando el campo para contraseña
Se dejan en blanco el nombre de usuario y la contraseña, los cuales son Required.	[16-06-2017] Se obtuvo el siguiente mensaje de error "Completa este campo" indicando el campo para usuario
Requerimientos asociados:	RF1

PA2: Inicio de sesión residente	
<b>Actor:</b> Residente	
<b>Precondición:</b> El residente recibirá un código (contraseña) para activar su usuario en la aplicación por única vez, el cual es solicitado al administrador. Además, la dirección de correo del usuario debe estar registrada en el celular como cuenta. Es Id del Usuario debe coincidir con esta dirección de correo.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
El residente recibe código de activación para loguearse al sistema Ingresa el código recibido por única vez. Se presiona el botón de ingreso El sistema acepta el código ingresado y el residente queda logueado al sistema.	[16-06-2017] Se inició sesión correctamente
<b>Cursos alternativos:</b>	
El residente ingresa un código de activación incorrecto, diferente al que le mandó el administrador.	[16-06-2017] Se obtuvo el siguiente mensaje de error: "Error de logueo: Reintente"
El residente no ingresa ningún código de activación, el cual es Required para loguearse.	[16-06-2017] Se obtuvo el siguiente mensaje de error: "Error de logueo: Reintente"
Requerimientos asociados:	RF1

PA3: Alta de anuncio
----------------------



<b>Actor:</b> Administrador - Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Administrador: Se completan los campos de texto. Se presiona el botón para publicar el anuncio. Residente Recibe notificación avisando de la publicación de un nuevo anuncio.	[16-06-2017] Anuncio creado correctamente y usuarios notificados
<b>Cursos alternativos:</b>	
Se deja vacío el campo de título	[16-06-2017] Se despliega el mensaje de error: ERROR: {"data": "The request is invalid."}
Si se dejó vacío el campo de cuerpo	[16-06-2017] Se despliega el mensaje de error: ERROR: {"data": "The request is invalid."}
Requerimientos asociados:	RF2

<b>PA4:</b> Baja de anuncios	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el anuncio a dar de baja. Se confirma la baja del anuncio.	[17-06-2017] El Anuncio se borró correctamente.
<b>Cursos alternativos:</b>	
No se selecciona ningún anuncio. Si no se seleccionó ningún anuncio, da un mensaje de error.	[17-06-2017] No es posible ya que cada anuncio cuenta con un botón particular para borrarlo.
Requerimientos asociados:	RF2

<b>PA5:</b> Alta de estado de cuenta	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Administrador: Se selecciona la opción de subir estado de	[17-06-2017] El Estado de Cuenta se creó correctamente.

cuenta. Ingresa el Usuario, Mes, Año e Importe. Se presiona el botón de confirmación para publicar el estado de cuenta.	
<b>Cursos alternativos:</b>	
No se ingresa cualquiera de los campos, todos ellos son requeridos.	[17-06-2017] Se despliega el mensaje de error: ERROR: {"data": "The request is invalid."}
Se ingresa un usuario que no existe.	[17-06-2017] Permite la creación del Estado de Cuenta aún cuando el usuario ingresado no existe.
Se ingresa un mes inválido.	[17-06-2017] Permite la creación del Estado de Cuenta aún cuando el mes ingresado no es válido.
Se ingresa un año inválido.	[17-06-2017] Permite la creación del Estado de Cuenta aún cuando el año ingresado no es válido.
Se ingresa un importe inválido.	[17-06-2017] Se obtuvo el siguiente mensaje de error: ERROR: {"data": "An error occurred while updating the entries. See the inner exception for details."}
Requerimientos asociados:	RF3

PA6: Baja de estado de cuenta	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el estado de cuenta a dar de baja. Se confirma la baja del estado de cuenta.	[17-06-2017] Se eliminaron todos los Estados de Cuenta en vez de únicamente el seleccionado.
<b>Cursos alternativos:</b>	
No se selecciona ningún estado de cuenta. Si no se seleccionó ningún estado de cuenta a dar de baja da un mensaje de error.	[17-06-2017] No es posible ya que cada Estado de Cuenta cuenta con un botón particular para eliminarlo.
Requerimientos asociados:	RF3

PA7: Control de asistencia en la reunión	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
Entrada	Salida
<p>Administrador</p> <p>A la hora del comienzo de la reunión se localiza a los clientes, contando cuántos de ellos están a menos de 30 metros del punto de reunión</p> <p>Se localiza a los residentes invitados.</p> <p>Si más del 50% de invitados está a menos de 30 metros de la zona de reunión, se da por comenzada la reunión.</p> <p>Residente</p> <p>Los residentes tienen habilitada la ubicación mediante el GPS.</p>	<p>[17-06-2017] Esta funcionalidad no se implementó ya que fue parte del cambio del alcance.</p>
<b>Cursos alternativos:</b>	
Si menos del 50% de los invitados está en esa zona, se espera hasta llegar a la cuota mínima de participantes.	
Requerimientos asociados:	RF4

PA8: Alta de tema de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
Entrada	Salida
<p>Se selecciona la opción de crear tema de votación.</p> <p>Se escribe el nuevo tema.</p> <p>Se escriben las opciones de voto para los residentes.</p> <p>Se confirma el nuevo tema de votación.</p>	<p>Esta operación se realiza directamente desde la Base de Datos.</p> <p>La gestión de Votaciones dentro de la página web no se encontraba dentro del alcance.</p>
<b>Cursos alternativos:</b>	
Se deja vacío el tema de votación.	
Se deja vacía la fecha de finalización.	
Requerimientos asociados:	RF14

PA9: Baja de tema de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de borrar tema de votación. Se selecciona el tema a borrar. Se confirma el borrado del tema de votación.	Esta operación se realiza directamente desde la Base de Datos. La gestión de Votaciones dentro de la página web no se encontraba dentro del alcance.
<b>Cursos alternativos:</b>	
No se selecciona ningún tema de votación.	
Si no se seleccionó ningún tema de votación, da un mensaje de error	
Requerimientos asociados:	RF4

PA10: Alta de tema de votación	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se ingresa la pregunta con formato de respuesta SI/NO. Se ingresa la fecha de finalización. Se confirma la votación.	[17-06-2017] La fecha de finalización se almacena con un mes antes del seleccionado.
<b>Cursos alternativos:</b>	
No se ingresa la pregunta con formato de respuesta SI/NO.	[17-06-2017] El tema de votación no se crea, pero no se despliega ningún mensaje de error.
No se ingresa la fecha de finalización.	[17-06-2017] No es posible ya que la fecha actual viene como fecha de finalización por default.
Se ingresa una fecha de finalización incorrecta.	[17-06-2017] Permite crear votaciones con fecha de finalización incorrecta.
Requerimientos asociados:	RF4

PA11:	Alta voto
-------	-----------

<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el tema de votación y se selecciona la opción que se desea votar.	[17-06-2017] Se ingresa el voto correctamente y se puede visualizar el cambio en el conteo de las votaciones.
<b>Cursos alternativos:</b>	
Se selecciona el tema de votación pero no se ingresa la opción a votar.	[17-06-2017] Se cancela la operación.
Requerimientos asociados:	RF4

PA12: Consultar voto	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se visualiza en la lista de votaciones la cantidad de votos.	[17-06-2017] Se visualiza el conteo de votos correctamente.
Requerimientos asociados:	RF4

PA13: Cierre de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Una vez alcanzada la fecha de finalización de la votación, no es posible realizar más votos.	[17-06-2017] Se permite ingresar votos incluso cuando el tema de votación debería encontrarse cerrado.
Requerimientos asociados:	RF4

PA14: Alta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el servicio a contratar. Se selecciona la fecha/hora de reserva. Se confirma la reserva.	[17-06-2017] La fecha de comienzo de la reserva se registra un mes antes.
<b>Cursos alternativos:</b>	

No se selecciona ningún servicio.	[17-06-2017] No es posible ya que el primer servicio viene seleccionado por default.
No se selecciona ninguna fecha.	[17-06-2017] No es posible ya que la fecha actual viene seleccionada por default.
Si se seleccionó una fecha que estaba ocupada, da mensaje de error.	[17-06-2017] Se obtiene el siguiente mensaje de error: "Se pisa con IdX Reserva XX"
Si se seleccionó una fecha anterior al día actual, da mensaje de error.	[17-06-2017] El mensaje de error no es amigable.
Requerimientos asociados:	RF6

PA15: Consulta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
Entrada	Salida
Se visualizan en el listado de reservar las reservas realizadas	[17-06-2017] Se visualiza la reserva correctamente.
Requerimientos asociados:	RF6

PA16: Alta de contacto en agenda	
<b>Actor:</b> Administrador	
<b>Precondición:</b> Puede existir más de un Contacto registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
Entrada	Salida
Se selecciona la opción de alta de contacto en agenda. Se escriben los datos del contacto, lo cual es mandatorio. Se confirma la creación del contacto.	Esta operación se realiza directamente desde la Base de Datos. La gestión de Contactos dentro de la página web no se encontraba dentro del alcance.
<b>Cursos alternativos:</b>	
Se deja algún dato del usuario en blanco.	
Si falta ingresar algún dato del usuario, da un mensaje de error.	
Requerimientos asociados:	RF7

PA17: Alta de contactos automática en agenda
--

<b>Actor:</b> Sistema	
<b>Precondición:</b> Puede existir más de un Contacto registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se crea un nuevo usuario en el sistema	[17-06-2017] Se crea un contacto automáticamente con la información del usuario.
Requerimientos asociados:	RF7

PA18: Modificación de usuario en agenda	
<b>Actor:</b> Administrador	
<b>Precondición:</b> Puede existir más de un usuario registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de modificación de usuario en agenda. Se modifican datos del usuario. Se confirma la modificación del usuario.	Esta operación se realiza directamente desde la Base de Datos. La gestión de Contactos dentro de la página web no se encontraba dentro del alcance.
<b>Cursos alternativos:</b>	
Se deja algún dato del usuario en blanco.	
Si falta ingresar algún dato del usuario, da un mensaje de error.	
Requerimientos asociados:	RF7

PA19: Baja de usuario en agenda	
<b>Actor:</b> Administrador	

<b>Precondición:</b> Puede existir más de un usuario registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de baja de usuario en agenda. Se selecciona el usuario a dar de baja. Se confirma la creación de usuario.	Esta operación se realiza directamente desde la Base de Datos. La gestión de Contactos dentro de la página web no se encontraba dentro del alcance.
<b>Cursos alternativos:</b>	
No se selecciona ningún usuario a dar de baja.	
Si no se seleccionó ningún usuario a dar de baja, da un mensaje de error.	
Requerimientos asociados:	RF7

PA20: Consulta de usuario en agenda	
<b>Actor:</b> Administrador – Residente	
<b>Precondición:</b> Puede existir más de un usuario registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se visualiza el listado de Contactos	[17-06-2017] La lista de Contactos se visualiza correctamente
Requerimientos asociados:	RF7

PA21: Compartir anuncio en Facebook	
<b>Actor:</b> Residente	



<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
El residente accede al anuncio y comparte el mismo logueándose en Facebook.	[17-06-2017] El anuncio se comparte en Facebook correctamente.
<b>Cursos alternativos:</b>	
1.El usuario no tiene cuenta en Facebook.	[17-06-2017] No es posible compartir el anuncio ya que no puede loguearse, a no ser que se registre en la aplicación.
Requerimientos asociados:	RF7

### 9.3.3 Pruebas de integración

Realizamos las pruebas de integración de forma de verificar el correcto funcionamiento de dichos elementos en forma conjunta. Básicamente buscamos probar la comunicación entre los distintos componentes(servidor, base de datos, sistemas, etc.). El método utilizado en este caso nuevamente es el de caja blanca(explicado en la parte anterior).

La idea es comprobar que interactúan correctamente, cubriendo las funcionalidades establecidas y ajustándose a los requisitos.

PI1: Alta de Usuario	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se completan los campos de usuario correctamente	Se ingresa una tupla a la tabla "Users" de la base de datos. Se ingresa una tupla a la tabla "Contacts" de la base de datos.
<b>Cursos alternativos:</b>	
Se completan los campos de usuario incorrectamente.	No se realiza ninguna modificación a nivel de base de datos.
Requerimientos asociados:	RF1

PI2: Inicio de sesión administrador	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	

<b>Entrada</b>	<b>Salida</b>
Se completan los campos de usuario y contraseña correctamente. Se inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
<b>Cursos alternativos:</b>	
Se completan los campos de usuario y contraseña correctamente, pero el nombre de usuario ingresado no existe en el sistema. No inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
Se completan los campos de usuario y contraseña correctamente, pero la password del usuario ingresado es incorrecta. No inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
Se completa el campo de usuario, pero el campo de password se deja vacío, y es Required. No inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
Se completa el campo de password, pero el campo de usuario se deja vacío, y es Required. No inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
Se dejan en blanco el nombre de usuario y la contraseña, los cuales son Required. No inicia sesión.	No se realiza ninguna modificación a nivel de base de datos.
Requerimientos asociados:	RF3

<b>PI3:</b>	<b>Inicio de sesión residente</b>
<b>Actor:</b>	Residente
<b>Precondición:</b>	El residente recibirá un código para activar su usuario en la aplicación por única vez, el cual es solicitado al administrador.
<b>Curso normal:</b>	

<b>Entrada</b>	<b>Salida</b>
El residente recibe código de activación para loguearse al sistema Ingresa el código recibido por única vez. Se presiona el botón de ingreso El sistema acepta el código ingresado y el residente queda logueado al sistema.	No se realiza ninguna modificación a nivel de base de datos.
<b>Cursos alternativos:</b>	
El residente ingresa un código de activación incorrecto, diferente al que le mandó el administrador.	No se realiza ninguna modificación a nivel de base de datos.
El sistema no acepta el código y el residente no puede ingresar al sistema.	No se realiza ninguna modificación a nivel de base de datos.
Requerimientos asociados:	RF1

<b>PI4: Alta de anuncio</b>	
<b>Actor:</b> Administrador - Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Administrador: Se completa el campo de texto y/o se adjunta/n alguna/s foto/s. Se presiona el botón para publicar el anuncio. Residente: Recibe notificación avisando de la publicación de un nuevo anuncio.	Se ingresa una tupla a la tabla "Anouncements" de la base de datos.
<b>Cursos alternativos:</b>	
Se deja vacío el campo de texto y no se adjunta ninguna imagen.	No se ingresa ninguna tupla en la tabla "Anouncements" de la base de datos.
Si se dejó vacío el campo de texto y no se adjuntó imagen, da mensaje de error.	No se ingresa ninguna tupla en la tabla "Anouncements" de la base de datos.
Requerimientos asociados:	RF2

<b>PI4: Baja de anuncios</b>	
<b>Actor:</b> Administrador	

<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el anuncio a dar de baja. Se confirma la baja del anuncio.	Se setea el campo "Deleted" de la tupla correspondiente en True.
<b>Cursos alternativos:</b>	
No se selecciona ningún anuncio. Si no se seleccionó ningún anuncio, da un mensaje de error.	No se realiza ningún cambio sobre la Base de Datos.
Requerimientos asociados:	RF2

PI5: Alta de estado de cuenta	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Administrador: Se selecciona la opción de subir estado de cuenta. Se selecciona el estado de cuenta a subir. Se presiona el botón de confirmación para publicar el estado de cuenta. Residente Recibe notificación avisando de la publicación del estado de cuenta.	Se inserta una tupla en la tabla "AccountStates" de la base de datos.
<b>Cursos alternativos:</b>	
No se selecciona ningún estado de cuenta. Si no se seleccionó ningún estado de cuenta, da un mensaje de error.	No se inserta ninguna tupla en la tabla "AccountStates" de la base de datos.
Requerimientos asociados:	RF3

PI6: Baja de estado de cuenta	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el estado de cuenta a	Se setea el campo "Deleted" de la tupla

dar de baja. Se confirma la baja del estado de cuenta.	correspondiente en True.
<b>Cursos alternativos:</b>	
No se selecciona ningún estado de cuenta. Si no se seleccionó ningún estado de cuenta a dar de baja da un mensaje de error.	No se realiza ningún cambio en la Base de Datos.
Requerimientos asociados:	RF3

PI7: Control de asistencia en la reunión	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
<p>Administrador A la hora del comienzo de la reunión se localiza a los clientes, contando cuántos de ellos están a menos de 30 metros del punto de reunión Se localiza a los residentes invitados. Si más del 50% de invitados está a menos de 30 metros de la zona de reunión, se da por comenzada la reunión. Residente Los residentes tienen habilitada la ubicación mediante el GPS.</p>	Esta funcionalidad no se implementó ya que fue parte del cambio del alcance.
<b>Cursos alternativos:</b>	
Si menos del 50% de los invitados está en esa zona, se espera hasta llegar a la cuota mínima de participantes.	
Requerimientos asociados:	RF4

PI8: Alta de tema de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se confirma el ingreso de un nuevo	Se inserta una tupla en la tabla "Votes" de la base

tema de votación.	de datos.
<b>Cursos alternativos:</b>	
Se deja vacío el nombre/objetivo del tema de votación.	No se inserta ninguna tupla en la tabla "Votes" de la base de datos.
Se escribe una única opción de voto para los residentes, da un mensaje de error	No se inserta ninguna tupla en la tabla "Votes" de la base de datos.
No se escriben opciones posibles de voto para los residentes, da un mensaje de error	No se inserta ninguna tupla en la tabla "Votes" de la base de datos.
Requerimientos asociados:	RF14

PI9: Baja de tema de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de borrar tema de votación. Se selecciona el tema a borrar. Se confirma el borrado del tema de votación.	Se setea el campo "Deleted" de la tupla correspondiente en True.
<b>Cursos alternativos:</b>	
No se selecciona ningún tema de votación.	No se realizan cambios en la Base de Datos
Requerimientos asociados:	RF4

PI10: Alta voto	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el tema por el cual votar.	Se realiza el UPDATE de la tupla correspondiente en la tabla "Votes" de la base de datos.

Se selecciona la opción de votación. Se confirma la votación.	
<b>Cursos alternativos:</b>	
No se selecciona ningún tema de votación.	No se realiza ningún cambios en la Base de Datos.
No hay temas de votación disponibles.	No se realiza ningún cambios en la Base de Datos.
No se selecciona ninguna opción de votación.	No se realiza ningún cambios en la Base de Datos.
Si no se seleccionó ningún tema de votación, da un mensaje de error	No se realiza ningún cambios en la Base de Datos.
Requerimientos asociados:	RF4

PI11: Consultar voto	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se visualiza el listado de Votaciones.	Se consultan todas las tuplas de la tabla "Votes" para mostrar el listado de Votaciones.
Requerimientos asociados:	RF4

PI12: Cierre de votación	
<b>Actor:</b> Administrador	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se alcanza la fecha de finalización y los usuarios no pueden ingresar más votos.	No se realizan modificaciones sobre la Base de Datos.
Requerimientos asociados:	RF4

PI13: Alta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el servicio a contratar.	Se realiza un INSERT de una tupla en la tabla

Se selecciona la fecha/hora de reserva. Se confirma la reserva.	"Bookings" de la base de datos.
<b>Cursos alternativos:</b>	
No se selecciona ningún servicio.	No se realiza ningún INSERT de tupla en la tabla "Bookings" de la base de datos.
No se selecciona ninguna fecha.	No se realiza ningún INSERT de tupla en la tabla "Bookings" de la base de datos.
Si se seleccionó una fecha que estaba ocupada, da mensaje de error.	No se realiza ningún INSERT de tupla en la tabla "Bookings" de la base de datos.
Si se seleccionó una fecha anterior al día actual, da mensaje de error.	No se realiza ningún INSERT de tupla en la tabla "Bookings" de la base de datos.
Requerimientos asociados:	RF6

PI14: Baja de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona el servicio contratado. Se selecciona la baja de la reserva. Se confirma la reserva.	Se setea el campo "Deleted" de la tupla correspondiente en True.
<b>Cursos alternativos:</b>	
No se selecciona ningún servicio.	No se realizan cambios sobre la Base de Datos.
No habían reservas efectuadas en ese servicio.	No se realizan cambios sobre la Base de Datos.
Requerimientos asociados:	RF6

PI15: Consulta de reserva	
<b>Actor:</b> Residente	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se visualiza el listado de Reservas	Se consultan todas las tuplas de la tabla "Bookings" para mostrar el listado de Votaciones.



Requerimientos asociados:	RF6
---------------------------	-----

PI16: Alta de contacto en agenda	
<b>Actor:</b> Administrador	
<b>Precondición:</b> Puede existir más de un usuario registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de alta de contacto en agenda. Se escriben los datos del contacto, lo cual es mandatorio. Se confirma la creación de contacto.	Se realiza un INSERT en la tabla "Contacts" de la Base de Datos.
<b>Cursos alternativos:</b>	
Se deja algún dato del usuario en blanco.	No se realiza ninguna modificación sobre la Base de Datos.
Si falta ingresar algún dato del usuario, da un mensaje de error.	No se realiza ninguna modificación sobre la Base de Datos.
Requerimientos asociados:	RF7

PI17: Baja de usuario en agenda	
<b>Actor:</b> Administrador	
<b>Precondición:</b> Puede existir más de un usuario registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se selecciona la opción de baja de usuario en agenda.	Se realiza un DELETE de la tupla correspondiente en la tabla "Contacts" de la Base de Datos.
<b>Cursos alternativos:</b>	
No se selecciona ningún usuario a dar de baja.	No se realizan cambios en la Base de Datos.
Si no se seleccionó ningún usuario a dar de baja, da un mensaje de error.	No se realizan cambios en la Base de Datos.
Requerimientos asociados:	RF7

PI18: Consulta de contacto en agenda	
<b>Actor:</b> Administrador	

<b>Precondición:</b> Puede existir más de un contacto registrado (como residente) por apartamento.	
<b>Curso normal:</b>	
<b>Entrada</b>	<b>Salida</b>
Se visualiza el listado de Contactos	Se consultan todas las tuplas de la tabla "Contacts" en la Base de Datos para listarlos.
Requerimientos asociados:	RF7

## 9.3 Estándares de documentación

Los estándares para realizar la documentación deben lograr un mantenimiento eficiente y diseño correcto de la misma. Debe ser instructiva de forma que una persona externa al proyecto (con mínimos conocimientos técnicos) pueda entender como funciona el sistema solamente con leer la documentación.

Consideramos importante que el estilo de redacción de los estándares de documentación cumpla con las siguientes características:

- Concreto.
- Ser preciso y definir los términos utilizados.
- Utilizar párrafos cortos.
- Utilizar títulos y subtítulos.
- Utilizar formas activas en lugar de pasivas.
- No usar frases largas que presenten hechos distintos.

Para realizar la presente documentación, nos basamos en los estándares contenidos en el Documento 302 de la Universidad ORT. En particular, en lo que detallan los capítulos: 1 (secciones 1.1, 1.5, 1.7, 1.8, 1.9), 4, 5 y 9.

# 10. Arquitectura

## 10.1 Diseño de la solución

En lo que refiere al diseño de la solución, la misma fue diseñada para implementarse en un solo edificio.

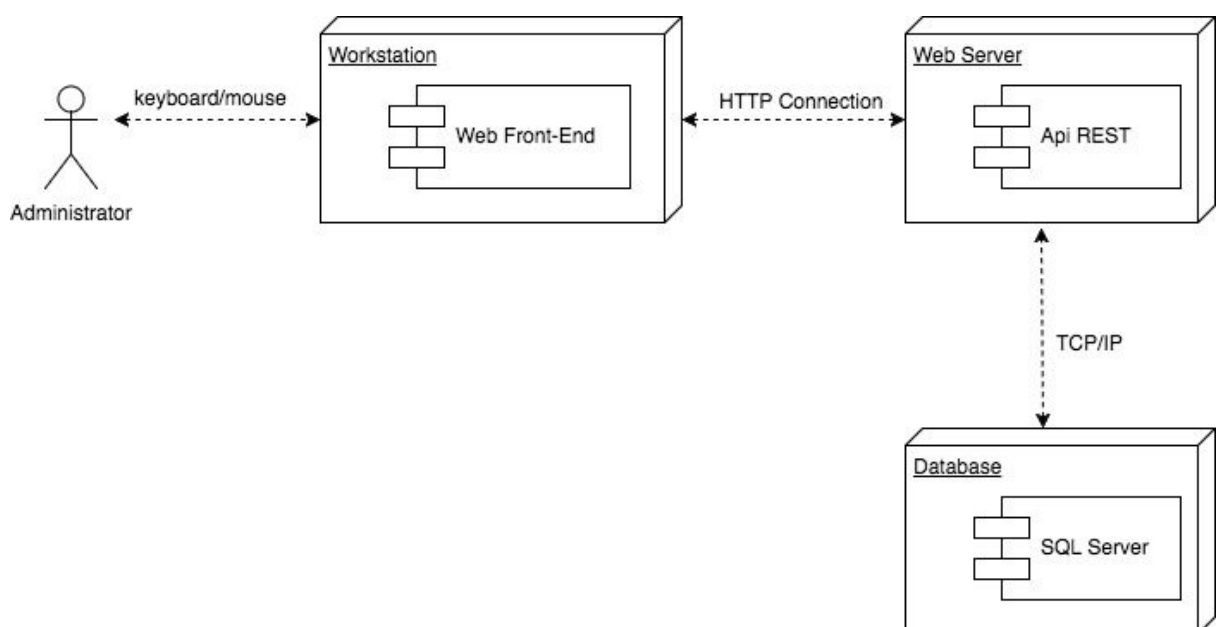
### 10.1.1 Arquitectura

La arquitectura es el diseño de más alto nivel de la estructura de un sistema. La misma se diseñó en base a requerimientos y restricciones establecidas por el cliente.

#### 10.1.1.2 Diagrama de componentes

De forma de expresar cómo se divide el sistema en componentes y mostrar cuales son las dependencias entre los mismos, es que realizamos los siguientes diagramas de componentes. Básicamente lo que buscamos con la realización de este diagrama es mostrar una vista de alto nivel de la solución planteada. El diagrama muestra claramente como son se comunican los componentes más genéricos del sistema y de qué forma se desarrolla el flujo de la aplicación.

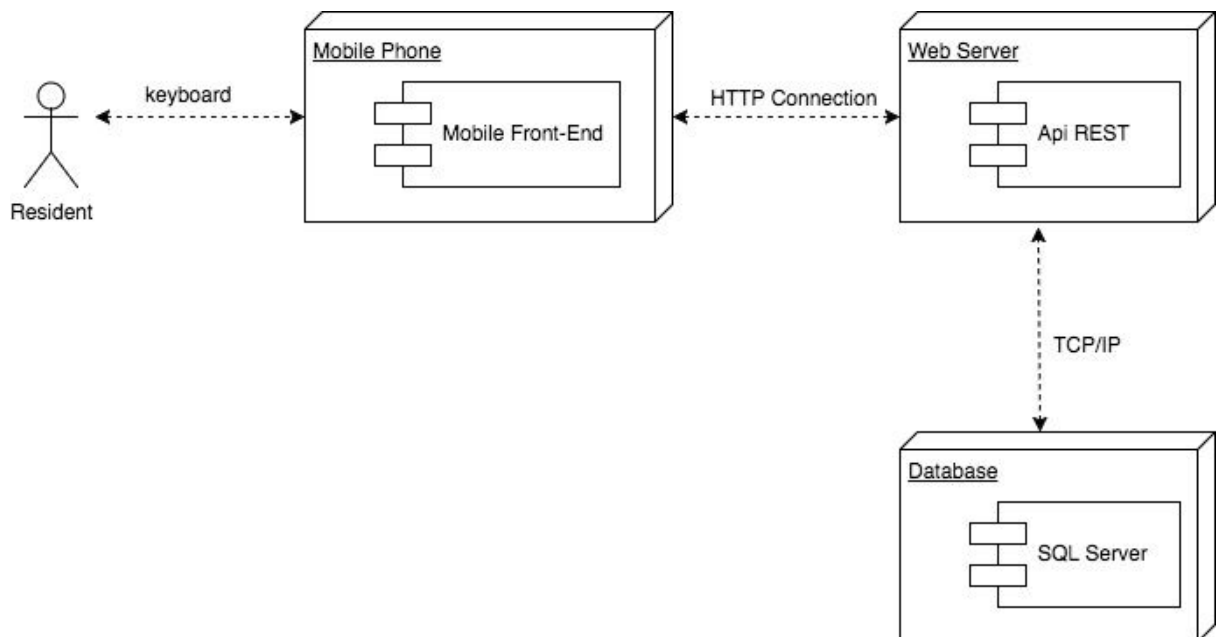
- El primero corresponde al usuario Administrador:



En particular, cada componente se define de la siguiente forma:

- El Web Front End es la aplicación que permite a los Administradores ingresar al sitio.
- La base de datos relacional SQL Server es en donde se almacenan los datos de la aplicación.
- La Api REST realiza toda la interacción con el repositorio de datos. Ofrece operaciones para resolver todo lo necesario.

- El segundo corresponde al Residente:



Para este caso, cada componente se define de la siguiente forma:

- El Mobile Front End, es la aplicación que permite a los Residentes ingresar en la aplicación.
- La base de datos relacional SQL Server es en donde se almacenan los datos de la aplicación.
- La Api REST realiza toda la interacción con el repositorio de datos. Ofrece operaciones para resolver todo lo necesario.

## 10.1.2 Diseño del FrontEnd

### Primera entrega

Para la primera entrega, realizamos los prototipos iniciales del diseño de interfaz de usuario. En esa instancia nos enfocamos en visualizar de cierta forma lo que será la interacción y contacto que el usuario establece a través de la computadora y el celular. Por este motivo, y de forma de obtener una guía visual sobre el diseño de la aplicación (en lo que refiere al sistema de navegación), construimos algunos mock-ups.

### Segunda entrega

Para la segunda entrega ya se realizó el maquetado final. El mismo se basó en los diseños de los mock-ups realizados para la primera entrega. Las consideraciones que tuvimos más en cuenta para este proceso fueron la legibilidad y usabilidad. En nuestro caso el maquetado está optimizado para Samsung gt-I8190L con Android 4.1.2.

Desarrollamos dos interfaces de usuario (Web y Mobile) utilizando la lógica del sistema común a ambas.

La página web se realizó utilizando Angular JS.

Por otro lado, la parte Mobile fue desarrollada utilizando Android Studio.

### 10.1.2.1 Mock-ups

La idea es acompañar el desarrollo del software, representando para los casos de uso más importantes, una idea de qué hace la pantalla y no de cómo se verá la misma. En este caso, llevamos a cabo un pequeño proceso de creación de mock-ups que consistió en:

- Realizamos a mano con lápiz y papel y de forma grupal varios prototipos para los casos de uso que se consideró más relevantes.

- Durante esta instancia se generó una discusión sobre cómo podría ser más usable o no cada prototipo.

- Unificamos los elementos que consideramos más relevantes, y construimos mediante la herramienta Balsamiq los modelos “finales”. Para esto, tuvimos en cuenta factores como: practicidad, usabilidad y navegabilidad de los mismos. Por ende, manejamos la posibilidad de que se realicen cambios a futuro en diferentes aspectos del diseño.

Los mock-ups contruidos se encuentran en la parte 10.1.1.1 Mockups de los Anexos.

## 10.1.2 Diseño del BackEnd

En la primera entrega no habíamos trabajado sobre el BackEnd, por lo que no hubo diferencias de implementación entre ambas entregas.

### 10.1.2.1 Almacenamiento de datos en la aplicación

Android provee varias formas de almacenar datos en una aplicación. En nuestro caso optamos por utilizar una denominada “Shared Preferences”. Ésta nos permite guardar y recuperar información de forma de : “clave:valor” . Es decir, podemos definir una clave unívoca a la cual asociamos el dato que queremos almacenar.

### 10.1.3 Justificaciones de Diseño

En esta sección se detallarán las principales decisiones tomadas a nivel de diseño de la aplicación, para cumplir con los requerimientos y lograr seguir una solución con alta cohesión, bajo acoplamiento, abierta a la extensión, etc.

Se cuenta con un componente llamado Data, el cual expone todas las entidades de negocio, que representan las abstracciones representadas a partir del modelo de negocio. El componente DataAccess contiene el acceso a base de datos y otro mecanismo de persistencia. Se utiliza Entity Framework (code first) y se aplica el patrón repositorio, el cual permite modificar la fuente de datos, minimizando el costo del cambio. Se supone que en una futura versión puede cambiar la manera de persistencia, o incluso convivir dos diferentes, y es por eso que se tomó la decisión de aplicar este patrón de diseño. La implementación se realizó con un repositorio genérico, el cual nos permite crear nuevas entidades sin necesidad de modificar nada en la persistencia. Para agregar una nueva entidad sin cambiar la base de datos, solo se debe agregar un objeto IRepository en la interface IUnitOfWork, y un nuevo atributo privado en la implementación de dicha interface.

Se manejaron excepciones propias, definidas en un componente separado, de manera de poder reutilizarlas en otro lugar y que todas las excepciones puedan ser utilizadas desde todos los componentes, sin necesidad de acoplarse a un componente solo por el uso de una excepción. Cada excepción recibe un mensaje personalizado que será escrito al momento de lanzar la excepción

Con el fin de manejar una única instancia de cada validador, se crea el componente Factory, el cual se encarga de manejar un Singleton de cada validador. De esta manera nos aseguramos que solo exista una instancia de cada validador, y que se creen en este componente. Si el día de mañana se decide cambiar la implementación de un validador, simplemente se cambia el new correspondiente en la clase SystemFactory, sin tener mayor impacto sobre el resto de los componentes.

El componente Logic es quien contiene las validaciones de los datos recibidos desde afuera de aplicación. Se decidió crear una interface para cada validador, de manera de poder cambiar la implementación en un futuro, logrando acoplarnos a la abstracción y no a la implementación. En esta primera versión se dejan las interfaces en el componente que las implementa, práctica que conocemos que no es la correcta, debiendo estar las interfaces, en un componente nuevo, o en componente cliente de la abstracción.

El componente Security es quién se encarga de validar que el usuario recibido en el header del request como usuario que se encuentra logueado sea válido. Para ello se recibe un Token.

En el componente Web.Api, se encuentran los controladores, los cuales reciben la información del request y la envían a la capa lógica, que se encarga de validar que los datos sean correctos. También en esta capa se “atrapan” todas las excepciones previstas y posibles

## 10.2 Modelo de Base de datos

### 10.2.1 Persistencia de Datos

Los datos de la solución se van almacenando durante la operativa de la misma, a medida que el usuario utiliza el sistema. Para dicha implementación, utilizamos el motor de base de datos Microsoft SQL Server 2012 Developer Edition.

El diseño contempla el modelado de una solución persistida utilizando Entity Framework mediante el enfoque Code First. El Entity Framework es un conjunto de APIs de acceso a datos para el Microsoft .NET Framework. Una entidad del Entity Framework es un objeto que tiene una clave representando la primary key (clave primaria) de una entidad lógica de datastore. Es un asignador objeto-relacional que nos permite trabajar con datos relacionales usando objetos específicos del dominio. Nos permite realizar un desarrollo más enfocado en el código, lo cual genera un flujo de desarrollo muy eficiente. Al utilizarlo, no tenemos la necesidad de definir un mala en XML ya que la interacción con la base de datos es mediante el programa.

En el caso del enfoque Code First, nos permite definir un modelo basado en crear en primera instancia el código y que después se genere automáticamente la base de datos.

Para interactuar con la base de datos, identificamos las clases del dominio que era necesario persistir y obtuvimos una tabla correspondiente en la base de datos para cada una de ellas. En las clases en las que contábamos con atributos que eran instancias de otras clases, al tratarse de relaciones 1-N representamos dichas instancias en el modelo de datos como FKs.

## 10.2.1 Modelo de tablas

El diagrama del modelo de tablas de la estructura de la base de datos se genera de forma automática por el motor SQL Server Management Studio, luego de creadas las tablas de la misma. En éste diagrama, se establecen las relaciones existentes entre las tablas, y por ende, los datos persistidos del programa. Estos datos pueden consultarse, hasta modificarse, siempre y cuando se realice desde el programa en Visual Studio, sino estaría rompiendo un principio fundamental de Code First.





### 10.2.2 Datos de Prueba

Creemos relevante la creación y el correcto uso de los datos de prueba, de forma que sean realistas y se adapten al propósito de nuestro software. Para eso, es importante considerar todas las entradas que nuestro sistema pudiese llegar a recibir.

Los objetivos más importantes a la hora de diseñar datos de prueba son:

- Detectar un error específico.
- Descubrir errores no descubiertos antes.
- Tener un buen caso de prueba.

En nuestro caso, creamos para cada tabla de la base de datos, un conjunto de datos que se insertarán previo a la prueba del software de forma de atacar posibles problemas futuros.

En los Anexos parte 10.2.2 Datos de prueba se adjuntan los datos de prueba para cada tabla.

# 11. Implementación

## 11.1 Código

### 11.1.1 Aplicación y evidencia de Clean Code

Tal como se plantea en la letra del obligatorio, tuvimos en cuenta que se realizarán futuras modificaciones en la aplicación, por lo que es fundamental que la misma tenga alto grado de mantenibilidad. Con el objetivo de realizar buenas prácticas al momento de escribir el código, utilizamos como base el libro Clean Code del autor Robert Martin. La idea fue adquirir herramientas que nos permitieran tener un código lo más comprensible posible.

Consideramos como aspecto primordial el hecho de leer detenidamente el código para llevar a cabo las técnicas que describe el libro. Como resultado de aplicar estas técnicas contamos con un código sencillo, fácil de leer y capaz de expresar el contexto del programa en el que se encuentra. Para obtener una mejor comprensión del código, se aplicaron algunas reglas en cuanto al formato que debe tener el mismo. Realizamos la separación de conceptos e ideas mediante espacios horizontales y verticales, y adoptamos la “Metáfora del Diario”. Esta última plantea, entre otras cosas, tener muchos métodos (artículos) de corta extensión, pudiendo existir alguno un poco más extenso.

Se menciona también la importancia que tienen los nombres de los métodos, variables, clases, etc. (los artículos en el caso del diario), ya que el mismo debería ubicarnos en la parte del código que deseamos sin que sea necesario comenzar a leer el contenido del método. Complementando esto, prestamos especial atención en algunos puntos respecto a los nombres:

- Se eligieron teniendo en cuenta que los mismos deben ser claros, legibles, mnemotécnicos, sencillos de pronunciar y lo suficientemente descriptivos.
- Tienen una única interpretación, que en caso contrario podría generar confusiones.
- Evitamos que los mismos sean verbos, por el contrario, al crear métodos si optamos por incluir un verbo en el nombre.
- Evitamos la utilización de atributos temporales, calculando los mismos en el momento, ya que los mismos no siempre tienen sentido todo el tiempo.

Para escribir las funciones, se tuvo en cuenta:

- Que las mismas sean de corta extensión (tomando como unidad de medida la cantidad de líneas de código).
- Que retornen algo o cumplan una única función,
- Se mantengan dentro del mismo nivel de abstracción.
- La indentación de cada línea en función de su ubicación dentro del código y dentro de cada método.

Se verificó la no duplicidad en el código escrito, así como la existencia de un número mínimo de argumentos: la cantidad ideal de argumentos en una función es cero. Además, tuvimos en cuenta que la presencia de un booleano como argumento de una función hubiera sido un error grave.

Al momento de analizar qué comentarios escribir(o no), tuvimos en cuenta siempre que los objetivos que estos se limitan básicamente a: resaltar la importancia de algo en particular, advertir sobre posibles consecuencias a otros programadores que accedan al código, etc. La idea fue transmitir mediante nuestro código todo lo que se pudiera transmitir, lo que nos ayudó a evitar la redundancia de comentarios.

Para el manejo de errores, buscamos mantener el código de manejo de errores de forma totalmente aislada de la lógica del programa. Esto permite mantener la comprensión del código sin que el manejo de errores se entreviera con la funcionalidad del mismo.

## 12. Manual de instalación

A continuación se detallan los pasos a seguir para poder instalar el sistema y poder utilizar tanto la aplicación Android como el backend desde la pag web.

1. Abrir la aplicación “VecinosUY.sln” en el Visual Studio
2. Acceder a la Aplicación web desde la URL: <http://104.155.174.201:45455/#/>  
Esta acción abrirá la página web con el backend.

Para loguearse en la misma deben utilizar las credenciales:

usuario: admin

contraseña: admin

**NOTA:** Esta URL no es accesible desde la red de ORT.

Si la página sigue inaccesible, por favor contactar a alguno de los miembros del equipo ya que el servidor público puede encontrarse fuera de servicio.

3. Correr la aplicación Android “App.VecinosUy” ya sea en un teléfono celular o un emulador.
4. Crear un nuevo usuario en el backend. El UserId debe coincidir con el correo electrónico registrado en el celular.  
En caso que el celular no tenga un correo electrónico asociado, el mismo debe configurarse a través de Configuración > Cuentas en el teléfono.

**NOTA:** Si la cuenta de correo electrónico no está creada en el teléfono correctamente o no coincide con el UserId de un usuario, no será posible realizar el login en la aplicación.

5. Ya que se trata de una cuenta de desarrollo, para poder probar la funcionalidad de compartir en Facebook deberán enviar a [facundo.laxalde@gmail.com](mailto:facundo.laxalde@gmail.com) los UserId utilizados para probar, para que los mismo puedan ser ingresados en la cuenta de desarrollo.

## 13. Referencias

Para el desarrollo del presente documento y el producto de software anteriormente descrito, se consultaron las siguientes fuentes de información:

- [1] Documentación de Estado de Cuenta de Gastos Comunes:  
Edificio Las Palmas  
Edificio Kendall Park
  
- [2] Proceso de Aseguramiento de la calidad y Gestión de proyectos:  
Libro: PMBOK - Fundamentos para la realización de proyectos 5ta edición - PMI
  
- [3] Clean Code:  
Libro: Clean Code - Robert C.Martin
  
- [4] Plan de comunicación:  
Libro: Guía de fundamentos para la dirección de proyectos 5ta edición - Estándar de Gerencia de proyectos establecido por el Project Management Institute (PMI).
  
- [5] Desarrollo Android:  
[https://www.tutorialspoint.com/android/android\\_shared\\_preferences.htm](https://www.tutorialspoint.com/android/android_shared_preferences.htm)
  
- [6] Desarrollo FrontEnd  
<https://desarrollofrontend.com/>
  
- [7] Realización de diagramas:  
<https://www.draw.io/>
  
- [8] Definiciones Glosario:  
<https://es.wikipedia.org/wiki/>  
<https://msdn.microsoft.com/es-es/>
  
- [9] Herramienta para realizar mock-ups:  
<https://balsamiq.com/>
  
- [10] Herramienta para realizar Gestión de Tareas y Control de cambios:  
<https://trello.com/>

## 14. Anexos

### 14.1 Actividades registrales

#### 14.1.1 Registro de esfuerzos

El registro de esfuerzos se encuentra documentado en la planilla que se encuentra en la siguiente URL:

<https://docs.google.com/spreadsheets/d/1efvHtWu6CJcagCe7uHonVqybS8mk5CqmmmMrJFngOck/edit#gid=0>

La totalidad de horas registradas fueron 152

#### 14.1.2 Estimaciones

El resultado de las estimaciones realizadas es el siguiente:

<b>Id</b>	<b>Requerimiento</b>	<b>BackEnd (hs)</b>	<b>FrontEnd (hs)</b>	<b>App (hs)</b>	<b>TOTAL (hs)</b>
RF1	Gestión de Usuarios Administradores	2	1	0	3
RF2	Gestión de Usuarios Residentes	2	0	1	3
RF3	Inicio de Sesión - Administradores	1	1	0	2
RF4	Inicio de Sesión - Residentes	1	0	1	2
RF5	Publicación de Anuncios	2	2	2	6
RF6	Visualización de Anuncios	1	0	2	3
RF7	Baja de Anuncios	1	1	0	2
RF8	Anuncios Favoritos	1	0	1	2
RF9	Compartir Anuncios	0	0	3	3
RF10	Publicación de Estados de Cuenta	1	1	0	2
RF11	Visualización de Estados de Cuenta	0	0	1	1

RF12	Coordinación de Reuniones de Residentes	1	1	1	3
RF13	Recordatorio de Reuniones de Residentes	0	0	4	4
RF14	Creación de Votaciones	3	3	3	9
RF15	Votación	0	0	2	2
RF16	Gestión de Salones o Servicios Comunes	2	2	0	4
RF17	Reserva de Salones o Servicios Comunes	2	0	2	4
RF18	Registro de Contactos	1	1	1	3
RF19	Agenda de Contactos	1	0	3	4

TOTAL DE HORAS ESTIMADAS (para requerimientos específicos): 62hrs

El total de horas ejecutadas fue mucho mayor al total estimado. Sin embargo, todo el esfuerzo registrado dedicado a la Documentación, a la familiarización de la tecnología, ajustes de estilos, entre otros no fueron estimados. Los mismos forman gran parte del exceso de horas.

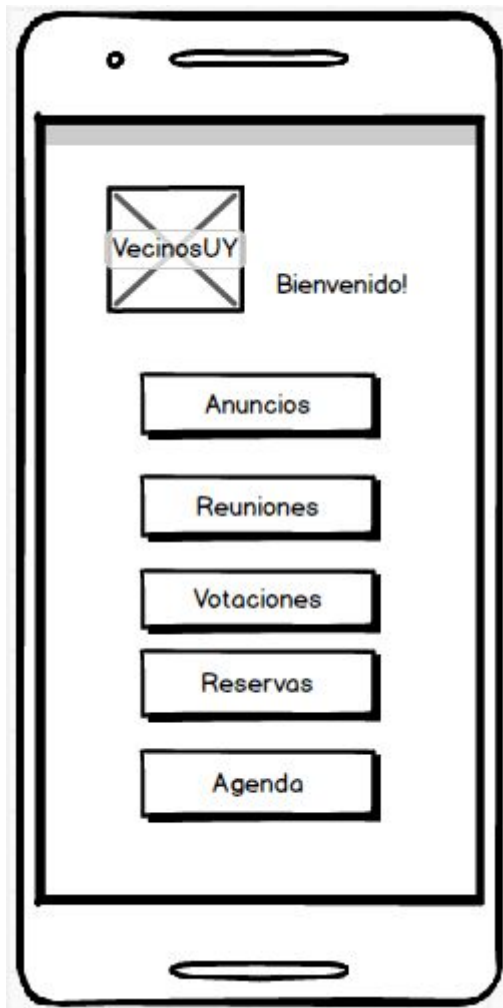
De todas formas, la implementación de los requerimientos también se desarrolló en más horas de las estimadas debido principalmente al desconocimiento de la tecnología. Sin embargo, el alcance de la entrega pudo realizarse de todas formas.

## 14.2 Mock-ups

### 14.2.1 Menú inicial residente

**Requerimiento funcional asociado:** RF2

**Objetivo:** Muestra el menú principal que le aparecerá al residente luego de ingresar al sistema desde la aplicación en su celular.

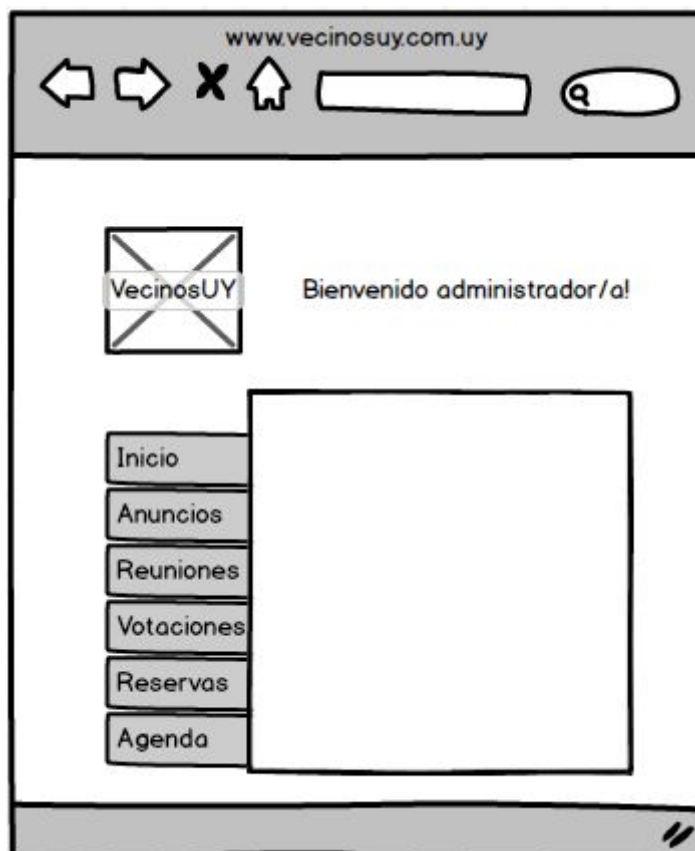




### 14.2.2 Menú inicial administrador

**Requerimiento funcional asociado:** RF1

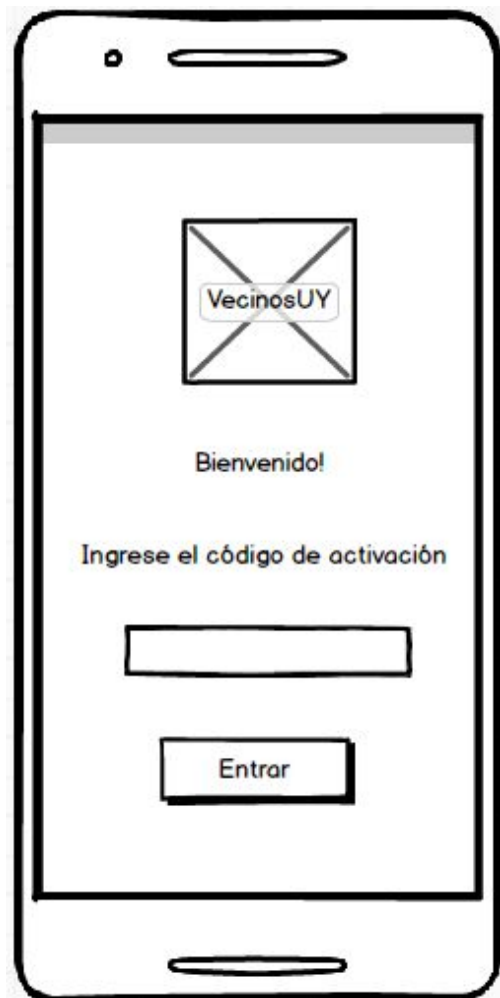
**Objetivo:** Muestra el menú principal que le aparecerá al administrador luego de ingresar al sistema desde la computadora.



### 14.2.3 Login residente

**Requerimiento funcional asociado:** RF4

**Objetivo:** Muestra la opción de loguearse por primera vez al residente desde la aplicación en su celular.



#### 14.2.4 Login administrador

**Requerimiento funcional asociado:** RF3

**Objetivo:** Muestra la pantalla que le aparecerá al administrador para loguearse al sistema desde la computadora.

The image is a hand-drawn sketch of a web browser window. The address bar at the top shows the URL "www.vecinosuy.com.uy". Below the address bar, there is a logo consisting of a square with an 'X' inside, and the text "VecinosUY" written across it. Below the logo, the text "Bienvenido al sitio de administradores!" is displayed. Underneath this, there are two input fields: the first is labeled "Usuario:" and the second is labeled "Contraseña:". To the right of the "Contraseña:" field is a button labeled "Entrar". The entire sketch is enclosed in a rectangular border representing the browser window.

### 14.2.5 Alta anuncio residente

**Requerimiento funcional asociado:** RF9

**Objetivo:** Muestra al residente la opción de dar de alta un anuncio.

The image shows a wireframe of a mobile application interface for creating a new announcement. The screen is titled "Inicio" with a back arrow icon. Below the title is the text "Crear nuevo anuncio". There are two input fields: "Titulo:" followed by a single-line text box, and "Contenido:" followed by a multi-line text box. Below these is a section for adding an image, labeled "Agregar imagen:", with a button labeled "Examinar". Below that is a section for opening the camera, labeled "Abrir cámara:", with a camera icon. At the bottom of the form is a button labeled "Publicar".

### 14.2.6 Alta anuncio administrador

**Requerimiento funcional asociado:** RF9

**Objetivo:** Muestra al administrador la opción de dar de alta un anuncio.

The image is a hand-drawn wireframe of a web browser window. The address bar at the top shows the URL "www.vecinosuy.com.uy". Below the address bar, there are navigation icons: a left arrow, a right arrow, a close button (X), and a home button (house icon). The main content area is titled "Crear nuevo anuncio". It contains two input fields: "Titulo:" followed by a single-line text box, and "Contenido:" followed by a multi-line text box. Below these fields, there is a label "Agregar imagen:" next to a button labeled "Examinar". Further down, there is a label "Abrir cámara:" next to a camera icon. At the bottom right of the form area, there is a button labeled "Publicar". The browser window has a grey border and a small icon in the bottom right corner.

### 14.2.7 Control de asistencia a reunión

**Requerimiento funcional asociado:** RF12

**Objetivo:** Muestra al administrador la cantidad de asistentes que hay cerca mediante el mapa.



#### 14.2.8 Alta reserva

**Requerimiento funcional asociado:** RF16, RF17.

**Objetivo:** Permite al residente reservar un servicio según la fecha solicitada.

The image shows a wireframe of a mobile application interface for reserving a service. The screen is enclosed in a rounded rectangle representing a smartphone. At the top, there is a header bar with a back arrow and the text "Inicio". Below this, the title "Reservar servicio" is centered. The main content area contains two sections: "Servicio:" followed by a list box with three items: "Item One", "Item Two", and "Item Three"; and "Fecha:" followed by a date input field with two slashes and a calendar icon. At the bottom, there is a button labeled "Confirmar reserva".

### 14.2.9 Cierre votación

**Requerimiento funcional asociado:** RF14, RF15

**Objetivo:** Permite al administrador dar finalización de un tema de votación, según la lista de temas que le aparezca.

