



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática
DOCUMENTACIÓN TÉCNICA



Presentado por Adrian Aguado
en Universidad de Burgos — 12 de junio de 2017
Tutor: Luis R.Izquierdo

Índice general

Índice general	I
Índice de figuras	II
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	7
Apéndice B Especificación de Requisitos	9
B.1. Introducción	9
B.2. Objetivos generales	10
B.3. Catálogo de requisitos	10
B.4. Especificación de requisitos	11
Apéndice C Especificación de diseño	25
C.1. Introducción	25
C.2. Diseño de datos	25
Apéndice D Documentación técnica de programación	26
D.1. Introducción	26
D.2. Requerimientos mínimos necesarios	27
D.3. AngularCLI	27
D.4. Manual del programador	29
Apéndice E Documentación de usuario	38
E.1. Introducción	38
E.2. Requisitos Usuarios	38
E.3. Primeros pasos	38

Índice de figuras

A.1. Detalle sprint 0	2
A.2. Detalle sprint 1	3
A.3. Detalle sprint 2	4
A.4. Detalle sprint 3	4
A.5. Detalle sprint 4	5
A.6. Detalle sprint 5	5
A.7. Detalle sprint 6	6
A.8. Detalle sprint 7	6
 B.1. Casos de uso. Fuente: Elaboración propia.	 11
 D.1. Comandos <i>AngularCLI</i> . Fuente: https://cli.angular.io/	 28
D.2. Consola ejecución. Fuente: http://codigoxules.org/	28
D.3. Resultado después de creado. Fuente: http://codigoxules.org/	29
D.4. Vista carpetas general <i>BarterAPP</i> . Fuente: Elaboración propia.	31
D.5. Vista Robomongo. Fuente: Elaboración propia.	36

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este capítulo se detalla la planificación del proyecto. Como gestor de tareas se comenzó utilizando *Trello+Github* pero más tarde se pasó a utilizar *Zenhub*, extensión de Google Chrome que permiten integrar los *boards* dentro del mismo repositorio de código alojado en *github*. Ya se han dado más detalles en la memoria del proyecto.

Se ha utilizado metodologías ágiles para el desarrollo del proyecto y de este modo, se ha realizado un desarrollo dividido en iteraciones. Terminada una iteración empezaba la siguiente y se agregaban a las tareas planeadas las que no habían sido completado de la iteración precedente. Las iteraciones del proyecto estaban pensadas para durar una diez días aproximadamente. No obstante, hay alguna excepción en la que la iteración duró más tiempo. También existe alguna demora entre algún sprint debido a que tenía demasiada carga de trabajo de las asignaturas, trabajaba o estaba de viaje.

La fase de planificación se puede dividir a su vez en:

- Planificación temporal.
- Estudio de viabilidad.

La primera parte me centro en la programación y desarrollo de la aplicación. Es decir elaboro un programa de tiempos con una serie de tareas a seguir para cumplimentar el proyecto.

La segunda parte se centra en el estudio de viabilidad. De la misma manera desde la segunda semana de marzo vengo realizando un plan de empresa con el programa Yuzz por lo que ello me va a facilitar el estudio de viabilidad de mi proyecto. Se desarrollará tanto la viabilidad legal como también la económica.

A.2. Planificación temporal

Desde inicio del proyecto se planteó utilizar una metodología ágil como *Scrum* para la gestión del proyecto. Aunque no se ha seguido al 100 % la metodología al tratarse de un proyecto para la Universidad, sí que se ha aplicado en líneas generales una filosofía ágil y metódica.

A continuación se describen los diferentes *sprints* que se han realizado. Dentro de *github* cada *milestone* recibe el número del sprint asignado y dentro de cada uno de ellos existen diferentes tareas que describiré a continuación. A cada tarea le acompaña un número a la derecha el cuál es denominado *story point*, de alguna manera sirve para realizar una estimación de lo que te va llevar completar esa determinada tarea. En mi caso concreto el 1 resulta ser el más bajo lo cuál indicaría que no más de dos o tres horas con cada tarea y el 13 el más alto lo cuál significa días de trabajo.

Sprint 0: 18/02/2017 - 28/02/2017

Tareas principales:

- Terminar formación.
- Aprender *Sonarqube*.
- Inicio Back-End.
- Inicio Decidir base de datos a emplear.

Completed Issues and Pull Requests	Story points
Terminar formación courses barterAPP #1 III New Issues ↑ Sprint 0	8
Aprender uso https://www.sonarqube.org/ research various barterAPP #2 III New Issues ↑ Sprint 0	1
Decidir base de datos a usar. research various barterAPP #3 III New Issues ↑ Sprint 0	1
Comenzar desarrollo Back-End. web app (Back-End) barterAPP #5 III New Issues ↑ Sprint 0	8
Información sobre bases de datos NoSQL courses research barterAPP #6 III New Issues ↑ Sprint 0	6

Figura A.1: Detalle sprint 0

La primera vez que hice uso de *ZenHub* ya llevaba algún tiempo formándome, de ahí el primer *Issue* del Sprint 0. Esta primera toma de contacto fue para comenzar a desarrollar el Back-End de la aplicación, además se consultaron varias fuentes para decidir que tipo de base de datos emplear.

Sprint 1: 18/02/2017 - 28/02/2017

Tareas principales:

- Corrección de errores en Back-End.
- Desarrollo Back-end.
- Errores en base de datos.

Completed Issues and Pull Requests			Story points
Actualizar Mendeley con el nuevo contenido consultado	documentation research		3
barterAPP #8	III New Issues	↑ Sprint 1	
Corregir errores en base de datos	bugs web app (Back-End)		5
barterAPP #9	III New Issues	↑ Sprint 1	
Añadir identificación usuarios	web app (Back-End)		8
barterAPP #10	III New Issues	↑ Sprint 1	
Actualizar package.json	web app (Back-End)		2
barterAPP #11	III New Issues	↑ Sprint 1	
Corregir errores entre versiones	bugs web app (Back-End)		5
barterAPP #12	III New Issues	↑ Sprint 1	
Errores en package.json	bugs web app (Back-End)		5
barterAPP #16	III New Issues	↑ Sprint 1	

Figura A.2: Detalle sprint 1

El Sprint 1 sirvió para continuar con el Back-end de la aplicación sin duda fue un de las partes más complicadas al pelearme con bases de datos con conceptos nuevos por lo que tuve numerosos bugs a la hora de guardar los usuarios en la base de datos.

Sprint 2: 15/03/2017 - 31/03/2017

Tareas principales:

- Corrección de errores en Back-End.
- Inicio Front-End.
- Bug en base de datos.
- Comienzo a leer sobre la documentación.

Completed Issues and Pull Requests	Story points
Landing-page web app (Front-End) barterAPP #13 III New Issues ↗ Sprint 2	5
Estructura visual web app (Front-End) barterAPP #15 III New Issues ↗ Sprint 2	3
MVC enhancement web app (Back-End) web app (Front-End) barterAPP #17 III New Issues ↗ Sprint 2	13
Bug sobre usuarios bugs web app (Back-End) barterAPP #18 III New Issues ↗ Sprint 2	5
Crear componentes en angular web app (Front-End) barterAPP #19 III New Issues ↗ Sprint 2	8
ReadTheDocs documentation enhancement barterAPP #20 III New Issues ↗ Sprint 2	5

Figura A.3: Detalle sprint 2

El Sprint 2 fue el momento donde una vez tenía un back-end sólido debía trasladarlo a la parte del usuario por lo que comencé a realizar la parte del front-end.

Sprint 3: 07/04/2017 - 15/04/2017

Tareas principales:

- Subir documentación.
- Elección del calendario.
- Corrección en componentes.

Completed Issues and Pull Requests	Story points
Publicar landing page en github pages bugs enhancement barterAPP #22 III New Issues ↗ Sprint 3	2
Subir /docs documentation barterAPP #23 III New Issues ↗ Sprint 3	1
Login de usuarios web app (Back-End) web app (Front-End) barterAPP #24 III New Issues ↗ Sprint 3	8
Crear nuevos usuarios en la app. web app (Back-End) web app (Front-End) barterAPP #25 III New Issues ↗ Sprint 3	6
Nuevo Schema para calendarios. web app (Front-End) barterAPP #26 III New Issues ↗ Sprint 3	3
Estudiar que calendario incluir dentro de la app enhancement research web app (Front-End) barterAPP #27 III New Issues ↗ Sprint 3	1
Corregir el fallo del servicio de angular2 bugs web app (Back-End) barterAPP #28 III New Issues ↗ Sprint 3	3

Figura A.4: Detalle sprint 3

El Sprint 3 se centra en la parte del calendario sobre todo, además de algo de documentación y corregir los errores que he arrastrado del back-end.

Sprint 4: 16/04/2017 - 22/04/2017

Tareas principales:

- Actualizar a Angular CLI.
- Heroku y MLab
- Documentación





Completed Issues and Pull Requests	Story points
 Actualizar a angular CLI enhancement web app (Back-End) web app (Front-End) barterAPP #29 III New Issues ↑ Sprint 4	(5)
 Crea base de datos online enhancement web app (Back-End) barterAPP #30 III New Issues ↑ Sprint 4	(1)
 Desplegar web en Heroku. enhancement barterAPP #31 III New Issues ↑ Sprint 4	(2)
 Mejorar documentación. documentation enhancement barterAPP #32 III New Issues ↑ Sprint 4	(2)

Figura A.5: Detalle sprint 4

El Sprint 4 es más corto dado que requiere un menor tiempo en realizar las tareas.

Sprint 5: 30/04/2017 - 07/05/2017

Tareas principales:

- Angular CLI.
- Bugs





Completed Issues and Pull Requests	Story points
 Continuar con la documentación documentation barterAPP #33 III New Issues ↑ Sprint 5	(3)
 Angular CLI vs Calendar bugs research web app (Front-End) barterAPP #34 III New Issues ↑ Sprint 5	(8)
 Error al guardar los elementos en la BD bugs web app (Front-End) barterAPP #35 III New Issues ↑ Sprint 5	(8)
 Angular CLI enhancement web app (Back-End) web app (Front-End) barterAPP #36 III New Issues ↑ Sprint 5	(3)

Figura A.6: Detalle sprint 5

El Sprint 5 fue complicado debido a que cambiar a Angular CLI resulta más sencillo a la hora de desplegar en servidor pero hay que saber como funciona realmente los proyectos en Angular CLI

Sprint 6: 09/05/2017 - 16/05/2017

Tareas principales:

- Tarea1
- Tarea2

Completed Issues and Pull Requests	Story points
<div><div><div></div><div>Error en turnos largos</div><div>bugs</div><div>web app (Back-End)</div><div>web app (Front-End)</div></div><div>barterAPP #37</div><div>III New Issues</div><div>⬆ Sprint 6</div></div>	6
<div><div><div></div><div>Anexos de la documentación.</div><div>documentation</div></div><div>barterAPP #38</div><div>III New Issues</div><div>⬆ Sprint 6</div></div>	2
<div><div><div></div><div>Mejorar aspecto visual.</div><div>enhancement</div><div>web app (Front-End)</div></div><div>barterAPP #39</div><div>III New Issues</div><div>⬆ Sprint 6</div></div>	13

Figura A.7: Detalle sprint 6

El Sprint 6

Sprint 7: 24/05/2017 - 31/05/2017

Tareas principales:

- Tarea1
- Tarea2

Completed Issues and Pull Requests	Story points
<div><div><div></div><div>Continuar documentación.</div><div>documentation</div></div><div>barterAPP #40</div><div>III New Issues</div><div>⬆ Sprint 7</div></div>	5
<div><div><div></div><div>Cambiar lógica de la app.</div><div>web app (Back-End)</div><div>web app (Front-End)</div></div><div>barterAPP #41</div><div>III New Issues</div><div>⬆ Sprint 7</div></div>	13
<div><div><div></div><div>Aplicar test para ver la calidad del código.</div><div>tests</div></div><div>barterAPP #42</div><div>III New Issues</div><div>⬆ Sprint 7</div></div>	5
<div><div><div></div><div>Leer libro "Clean Code"</div><div>research</div><div>various</div></div><div>barterAPP #43</div><div>III New Issues</div><div>⬆ Sprint 7</div></div>	Not estimated

Figura A.8: Detalle sprint 7

El Sprint 7

Sprint 8: 01/06/2017 - 10/06/2017

Tareas principales:

- Tarea1

- Tarea2

El Sprint 8

Sprint 9: 10/06/2017 - 20/06/2017

Tareas principales:

- Tarea1
- Tarea2

El Sprint 9

Sprint 10: 20/06/2017 - 30/06/2017

Tareas principales:

- Tarea1
- Tarea2

El Sprint 10

A.3. Estudio de viabilidad

En esta sección se lleva a cabo un estudio para comprobar la viabilidad del proyecto realizado. Paralelamente al desarrollo de la aplicación, como ya se ha nombrado en la memoria anteriormente, el proyecto formó parte del programa YUZZ para jóvenes emprendedores en el que durante cinco meses realicé un plan de empresa completo. Se detalla por tanto en un documento que adjuntaré al proyecto un estudio de viabilidad exhaustivo y muy completo en el que se incluyen entre otras cosas: plan de marketing, plan de financiación, estudio de viabilidad o plan de puesta en marcha del negocio a cinco años vista.

Por lo tanto en esta sección voy a realizar un resumen del documento descrito en el párrafo anterior en el que como conclusión definitiva tendremos un boceto de lo que supondría transformar un proyecto fin de carrera y que pase a formar parte del mercado. Así mismo voy a intentar adaptarlo a las condiciones que se exigen en el proyecto dado que el plan de empresa completo es un estudio de viabilidad completo de aquí a cinco años por lo que resulta ser más extenso y detallado. Se intentará por tanto aquí realizar una estimación.

Viabilidad económica

La viabilidad económica es la parte donde lograremos detectar si el proyecto es o no rentable económicamente hablando.

Análisis de costes

Económica

Coste de personal Se considerará que el proyecto ha sido desarrollado en un periodo de cinco meses. Considerando que se ha trabajado unas 6 horas a día cada semana, y que el programador, que en este caso es una sola persona, ha percibido un sueldo de 13 e/hora, el coste del personal por lo tanto se resume en la siguiente tabla:

	Total
13 €/hora * 6 horas/día	78 €/día
78 €/día * 5 días/semana	390 €/semana
390 €/semana * 4 semanas/mes	1560 €/mes
Coste total salario	7800 €/5 meses

Tabla A.1: Tabla salarios.

Coste de seguridad social más información sobre el segundo item.

Coste de software más información sobre el segundo item.

Coste de Hardware más información sobre el segundo item.

Viabilidad legal

La viabilidad legal se centra principalmente en el estudio de las licencias software utilizadas y en la licencia que se le va a ser asignada a las diferentes aplicaciones desarrolladas.

Especificación de Requisitos

B.1. Introducción

Este anexo recoge la especificación de requisitos que define el comportamiento del sistema desarrollado. El objetivo principal de la Especificación de Requisitos del Sistema (*ERS*) es servir como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores.

La ERS es correcta si y sólo si todo requisito que figura en ella refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el sistema deseado. Se han seguido las recomendaciones del estándar IEEE 830 según la última versión del estándar IEEE 830. Las características deseables para una especificación de requisitos son:

1. No ambigua.
2. Completa
3. Verificable
4. Consistente
5. Clasificada
6. Modificable
7. Explorable
8. Utilizable durante las tareas de mantenimiento y uso

B.2. Objetivos generales

Los objetivos generales que se perseguían con el proyecto han sido:

- Desarrollar una aplicación web que permita intercambiar turnos en función de unas restricciones.
- Desarrollar una aplicación híbrida que adapte la anterior funcionalidad a cualquier dispositivo.

B.3. Catálogo de requisitos

A continuación, se enumeran los requisitos específicos :

En esta sección se especificarán los requisitos del sistema, diferenciando los requisitos funcionales, o sea los el comportamiento del sistema, de los requisitos no funcionales, que describen características de funcionamiento.

Requisitos funcionales

RF-1 Gestión de usuarios: Los usuarios son la principal baza de la aplicación se deben poder gestionar correctamente.

- **RF-1.1:** Registrar usuarios
- **RF-1.2:** Logear usuarios
- **RF-1.3:** Deslogearse

RF-2. Gestión de calendario: Funciona como un calendario normal.

- **RF-2.1:** Añadir turnos.
- **RF-2.2:** Eliminar turnos.
- **RF-2.3:** Editar turnos.
- **RF-2.4:** Visualizar turnos.

RF-3. Gestión de cambios: Parte de intercambio de turnos.

- **RF-3.1:** Enviar petición.
- **RF-3.2:** Aceptar petición.
- **RF-3.3:** Rechazar petición.
- **RF-3.4:** Aceptar intercambio.
- **RF-3.5:** Rechazar intercambio.
- **RF-3.6:** Visualizar detalles.
 - **RF-3.6.1:** Visualizar turnos.

- **RF-3.6.2:** Visualizar pendientes.
- **RF-3.6.3:** Visualizar aceptados.
- **RF-3.6.4:** Visualizar rechazados.
- **RF-3.7:** Visualizar calendario.

RF-4. Información: Información de como usar la aplicación

Requisitos no funcionales

RNF-1 Seguridad texto.

RNF.2 Escalabilidad: texto

RNF.3 Eficiencia: texto.

B.4. Especificación de requisitos

En esta sección se explicará el diagrama de casos de uso y se desarrollará cada uno de los requisitos en función del esquema.

Diagrama casos de uso

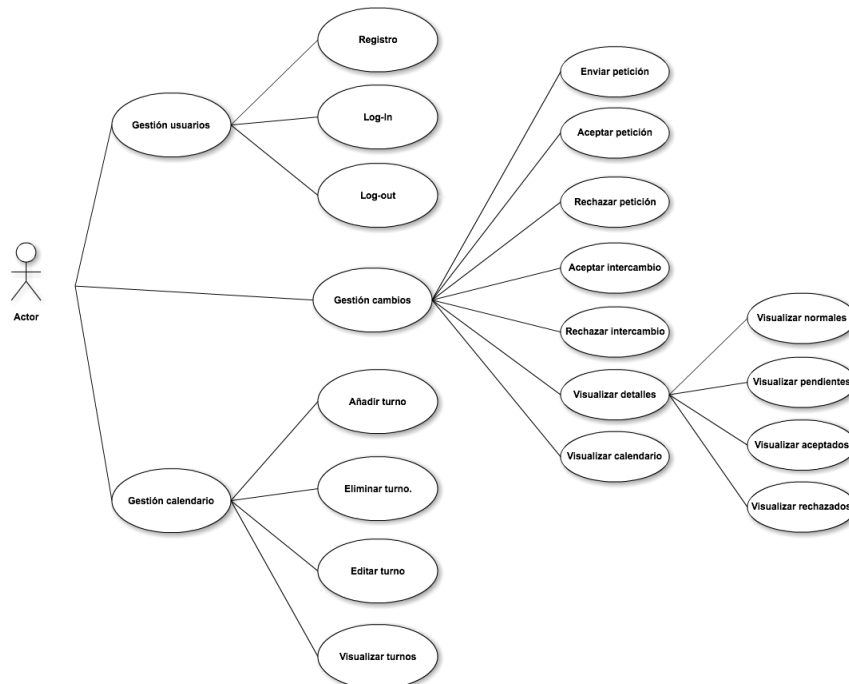


Figura B.1: Casos de uso. Fuente: Elaboración propia.

CU-01	Gestión usuarios
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-1.1, RF-1.2, RF-1.3
Descripción	Permite registrarse, logearse al usuario o salir.
Precondición	La página está cargada, base de datos funcionando.
Acciones posibles	1. El usuario se registra. 2. El usuario se logea. 3. El usuario interactúa. 4. El usuario sale.
Postcondición	Se añade el usuario a la base de datos.
Excepciones	
Frecuencia	Alta
Importancia	Alta
Comentarios	

Tabla B.1: CU-0 Gestión usuarios

Actores

El actor es el usuario de la aplicación, tenemos que tener en cuenta que para que se de un cambio de turno deben de existir dos actores, sino sería imposible el intercambio.

Casos de uso

Nota: El editar usuarios y eliminar usuarios no esta implementado de cara al usuario. Esta hecho pero no se encuentra disponible en pantalla para realizarse directamente por parte del usuario.

CU-02	Registro
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-1.1
Descripción	Permite registrarse.
Precondición	La página está cargada en la pestaña adecuada.
Acciones	1. El usuario introduce los datos, a saber: A. Nombre B. Apellidos B. Nombre usuario C. E-mail D. Nombre empresa E. Tipo de turno (a elegir) F. Contraseña
Postcondición	Se añade el usuario a la base de datos. Redirección a log-in.
Excepciones	Si no se introducen todos los campos obligatorios se notifica. También si el usuario ya existe o si todo ha ido correctamente.
Frecuencia	Alta
Importancia	Alta
Comentarios	

Tabla B.2: CU-02 Registro

CU-03	Log-in
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-1.2
Descripción	Permite logearse al usuario
Precondición	La página está cargada en la pestaña adecuada. Base de datos funcionando.
Acciones	1. El usuario se <i>logea</i> para ellos introduce: A. Correo electrónico B. Contraseña 2. Se cargan los turnos en el calendario.
Postcondición	El usuario entra en la aplicación.
Excepciones	Si la contraseña no es correcta, aviso. Si el correo no es correcto, aviso.
Frecuencia	Alta
Importancia	Alta
Comentarios	Si fuera el primer <i>log-in</i> el calendario el calendario estará vacío.

Tabla B.3: CU-03 Log-in

CU-04	Log-out
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-1.3
Descripción	Permite salir de la aplicación.
Precondición	El usuario está dentro de la aplicación.
Acciones	1. Presionar el botón <i>log out</i> .
Postcondición	Sale de la aplicación.
Excepciones	
Frecuencia	Alta
Importancia	Alta
Comentarios	

Tabla B.4: CU-04 Log out

CU-05	Gestión calendario
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-2.1, RF-2.2, RF-2.3, RF-2.4
Descripción	Gestión del calendario.
Precondición	El usuario está dentro de la aplicación.
Acciones	<ol style="list-style-type: none"> 1. Añadir turnos. 2. Eliminar turnos. 3. Editar turnos. 4. Visualizar turnos.
Postcondición	Turno deseado.
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	

Tabla B.5: CU-05 Gestión Calendario

CU-06	Añadir turno
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-2.1
Descripción	Añadir un turno determinado.
Precondición	El usuario está dentro de la aplicación.
Acciones	<ol style="list-style-type: none"> 1. Presionar botón <i>nuevo turno</i>. 2. Seleccionar título. 3. Seleccionar tipo. 4. Seleccionar fecha.
Postcondiciones	<p>Turno deseado en calendario.</p> <p>Turno añadido en la base de datos asociado al usuario que lo ha añadido.</p> <p>Prioridad normal.</p>
Excepciones	No es posible más de un turno al día.
Frecuencia	Media
Importancia	Alta
Comentarios	Se controla que solo se añada un turno al día. No se controla, todavía, en que franja horaria se realizan.

Tabla B.6: CU-06 Añadir turno

CU-07	Eliminar turno
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-2.3
Descripción	Eliminar un turno.
Precondición	El usuario está dentro de la aplicación.
Acciones	1. Presionar botón <i>eliminar turno</i> 2. Turno eliminado.
Postcondiciones	Turno eliminado del calendario. Turno eliminado de la base de datos.
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	

Tabla B.7: CU-07 Eliminar turno

CU-08	Editar turno
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-2.3
Descripción	Editar un turno.
Precondición	El usuario está dentro de la aplicación.
Acciones	1.El usuario visualiza el turno en la pestaña <i>normal</i> . 2. El usuario puede modificar : A.Titulo B.Tipo C.Fecha
Postcondición	Turno deseado editado.
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	El usuario puede modificar un turno siempre que así lo desee.

Tabla B.8: CU-08 Editar turno

CU-09	Visualizar turnos
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-2.4
Descripción	Ver de un simple vistazo todos los turnos.
Precondición	El usuario está dentro de la aplicación.
Acciones	1. Ver turnos.
Postcondición	
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	

Tabla B.9: CU-09 Visualizar turnos

CU-10	Gestión cambios
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.1, RF-3.2, RF-3.3, RF-3.4 RF-3.5, RF-3.6, RF-3.6.1, RF-3.6.2 RF-3.6.3, RF-3.6.4, RF-3.7
Descripción	Gestión de los cambios entre usuarios.
Precondición	El usuario está dentro de la aplicación.
Acciones	Hay más de un usuario. 1. Enviar petición de cambio. 2. Aceptar petición de cambio. 3. Rechazar petición de cambio. 4. Aceptar intercambio. 5. Rechazar intercambio. 6. Visualizar detalles. 7. Visualizar calendario.
Postcondición	Cambio turno.
Excepciones	Para diversos casos se muestran avisos.
Frecuencia	Media
Importancia	Alta
Comentarios	Se controlan que la petición se envíe una sola vez. Se muestra la lista de usuarios con turnos libres asociada a un día concreto

Tabla B.10: CU-10 Gestión Cambios

CU-11	Envío de petición de cambio
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.1
Descripción	Gestión de los turnos entre usuarios.
Precondición	El usuario está dentro de la aplicación. Hay más de un usuario.
Acciones	1. Enviar petición de cambio.
Postcondición	Ver si el usuario . destinatario a aceptado o rechazado la petición.
Excepciones	Aviso de envío de petición tan solo una vez. Si no hay usuarios libres no es posible enviar la petición
Frecuencia	Media
Importancia	Alta
Comentarios	Se controlan que la petición se envíe una sola vez. Se tienen en cuenta todos los usuarios con turnos libre.

Tabla B.11: CU-11 Enviar petición

CU-12	Aceptar de petición de cambio
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.1
Descripción	Aceptación de la petición
Precondición	El usuario está dentro de la aplicación. Hay una petición por parte de otro usuario
Acciones	1. Aceptar petición.
Postcondición	Cambio aceptado.
Excepciones	Petición única
Frecuencia	Media
Importancia	Alta
Comentarios	Se controlan que la petición sea única.

Tabla B.12: CU-12 Aceptar petición

CU-13	Rechazar de petición de cambio
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.2
Descripción	Rechazo de la petición
Precondición	El usuario está dentro de la aplicación. Hay una petición por parte de otro usuario
Acciones	1. Rechazar petición.
Postcondición	Fin de la lógica.
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	Se controlan que la petición sea única.

Tabla B.13: CU-13 Rechazar petición

CU-13	Aceptar intercambio
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.3
Descripción	Aceptación intercambio
Precondición	El usuario está dentro de la aplicación. Hay una petición por parte de otro usuario
Acciones	1. Aceptar cambio.
Postcondición	Se produce el cambio. Los eventos se intercambian tanto en la base de datos como en el calendario de ambos usuarios
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	

Tabla B.14: CU-14 Aceptar intercambio

CU-15	Rechazar de petición de cambio
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.4
Descripción	Rechazo de la petición
Precondición	El usuario está dentro de la aplicación. Hay una petición por parte de otro usuario
Acciones	1. Rechazar intercambio.
Postcondición	Fin de la lógica.
Excepciones	
Frecuencia	Media
Importancia	Alta
Comentarios	

Tabla B.15: CU-15: Rechazar intercambio

CU-16	Detalles peticiones
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6, RF-3.6.1 RF-3.6.2, RF-3.6.3, RF-3.6.4
Descripción	Detalles de las peticiones mediante botones para una mejor comprensión por parte del usuario
Precondición	El usuario está dentro de la aplicación.
Acciones	1. Visualizar normales. 2. Visualizar pendientes. 3. Visualizar aceptados. 4. Visualizar rechazados.
Postcondición	Ver numero de cada parámetro
Excepciones	
Frecuencia	Media
Importancia	Media
Comentarios	No es estrictamente necesaria esta parte pero ayuda al usuario

Tabla B.16: CU-16: Visualizar detalles

CU-16	Detalles turnos
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6.1
Descripción	Detalle de turnos
Precondición	El usuario está dentro de la aplicación y ha realizado algún turno
Acciones	1. Visualizar normales.
Postcondición	
Excepciones	
Frecuencia	Media
Importancia	Media
Comentarios	

Tabla B.17: CU-17: Visualizar normales

CU-16	Detalles peticiones pendientes
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6.2
Descripción	Detalle de peticiones pendientes
Precondición	El usuario está dentro de la aplicación y ha solicitado algún cambio de turno (petición pendiente)
Acciones	1. Visualizar peticiones pendientes.
Postcondición	
Excepciones	
Frecuencia	Media
Importancia	Media
Comentarios	

Tabla B.18: CU-18 Visualizar pendientes

CU-16	Detalles peticiones
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6.3
Descripción	Detalle de peticiones aceptadas
Precondición	El usuario está dentro de la aplicación y ha realizado alguna petición de cambio.
Acciones	1. Visualizar peticiones aceptadas .
Postcondición	Confirmar cambio
Excepciones	Si la petición se ha rechazado no se visualizará aquí.
Frecuencia	Media
Importancia	Media
Comentarios	

Tabla B.19: CU-17: Visualizar normales

CU-20	Detalles peticiones
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6.4
Descripción	Detalle de peticiones rechazadas
Precondición	El usuario está dentro de la aplicación y ha realizado alguna petición de cambio.
Acciones	1. Visualizar peticiones rechazadas .
Postcondición	Rechazar cambio
Excepciones	Si la petición se ha aceptado no se visualizará aquí.
Frecuencia	Media
Importancia	Media
Comentarios	

Tabla B.20: CU-20 Visualizar rechazadas

CU-20	Visualizar calendario
Versión	1.0
Autor	Adrián Aguado
Requisitos asociados	RF-3.6.4
Descripción	Visualizar cambios en el calendario
Precondición	El usuario está dentro de la aplicación.
Acciones	1. Visualizar turnos 2. Click encima de turnos para ver duración 3, Color asociado a cada tipo de turno
Postcondición	
Frecuencia	Media
Importancia	Media
Comentarios	Todos los cambios de turno se deben de ver reflejados en el calendario del usuario en caso de ser aceptados por ambas partes

Tabla B.21: CU-21 Visualizar calendario

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado se incluye el diseño de la aplicación y se mostrarán también los diagramas de clases de cada una de las partes que forman el proyecto.

C.2. Diseño de datos

En este apartado se incluye el diseño de la aplicación y se mostrarán también los diagramas de clases de cada una de las partes que forman el proyecto.

Documentación técnica de programación

D.1. Introducción

En este capítulo vamos a dividirlo en dos partes, por un lado los condiciones específicas propias de la aplicación y por otro vamos a aprender de manera breve y sencilla a crear una aplicación gracias a *Angular CLI*, la interfaz de línea de comandos de Angular.

El por qué he decidido incluir la parte de *Angular CLI* es por que considero que va aclarar muchas dudas acerca de cómo funciona esta tecnología y también para intentar comprender mejor como he realizado el proyecto. En diversas ocasiones para aprender a usar una tecnología tan solo hace falta tiempo pero en otras muchas el tiempo es limitado por lo que quizás con esta pequeña guía podemos comenzar a utilizar un framework cliente en muy pocos pasos y de una manera sencilla. Si bien es cierto que yo he necesitado mucho tiempo para comprender su funcionamiento y no lo he aprendido todo gracias a la interfaz de comandos, ésta te permite agilizar los trámites de creación en un tanto por ciento considerablemente alto.

Este anexo por tanto tiene como objetivo analizar y documentar las necesidades funcionales que deberán ser soportadas por el sistema a desarrollar, es decir, en qué condiciones ha sido desarrollado, en qué condiciones se debe usar y cuáles son los requerimientos mínimos para que un futuro programador interactúe con la aplicación en caso de que así lo considere.

D.2. Requerimientos mínimos necesarios

Además de un *IDE* para poder trabajar, los prerequisites mínimos para empezar a trabajar en el proyecto son:

1. Instalar Nodejs y MongoDB.
2. Instalar npm
3. Instalar Angular CLI

```
npm i -g @angular/cli
```

Nota: es interesante, una vez instalados asegurarnos de que están correctamente instalados usando los comandos `node -v` y `npm -v` para saber si tenemos los requerimientos correctamente instalados.

D.3. AngularCLI

Sabiendo los requisitos mínimos que son necesarios para proceder vamos a realizar, cómo he nombrado antes una breve introducción a la línea de comandos de Angular, **Angular CLI**.

¿Qué es typescript?

Esto se ha nombrado antes pero resulta esencial para entender el código de Angular. Antes de nada decir que en Angular2 se puede trabajar con Javascript “clásico” (ES5), así como ES6 y TypeScript. Desarrollado por Microsoft y como apuesta de Google vamos a ver el crecimiento exponencial que ha tenido este de este lenguaje, que no deja de ser un super conjunto de JavaScript. TypeScript es por tanto un superset de ECMAScript 6, es decir, incluye todas las funcionalidades de ES6, y además incorpora una capa por encima con funcionalidades extra.

*TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
Any browser. Any host. Any OS. Open source.*

([?])

La principal característica de TypeScript por encima de Javascript es que permite definir de qué tipo son las variables que se van a usar.

Comandos básicos

Angular CLI nos va a hacer la vida más fácil con Angular, ya que nos permite crear de forma sencilla un aplicación lista para funcionar después de la instalación por línea de comandos con *npm*, además incorpora las buenas prácticas de programación con Angular.

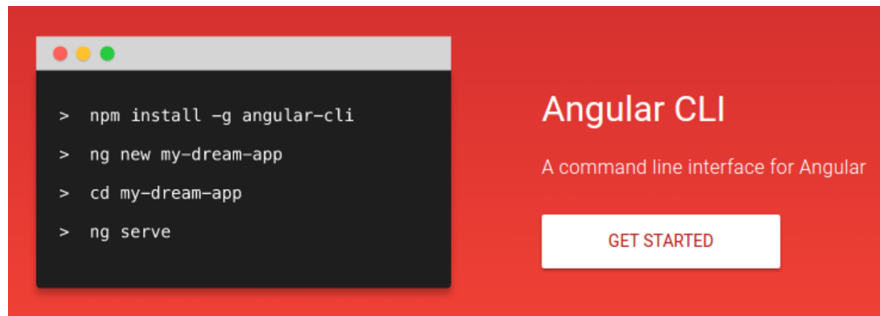


Figura D.1: Comandos *AngularCLI*. Fuente: <https://cli.angular.io/>.

Por lo tanto para crear nuestra aplicación con esta herramienta tan solo debemos ir a nuestra carpeta donde queremos crear el proyecto e introducir en la línea de comandos:

```
ng new angular-cli-primer-proyecto
```

Es un proceso que durará unos 2 minutos, Al finalizar seremos capaces de ejecutar directamente la aplicación por defecto que crea la herramienta. Para ejecutarla, al igual que antes accedemos a la carpeta del proyecto y ejecutamos:

```
npm start
```

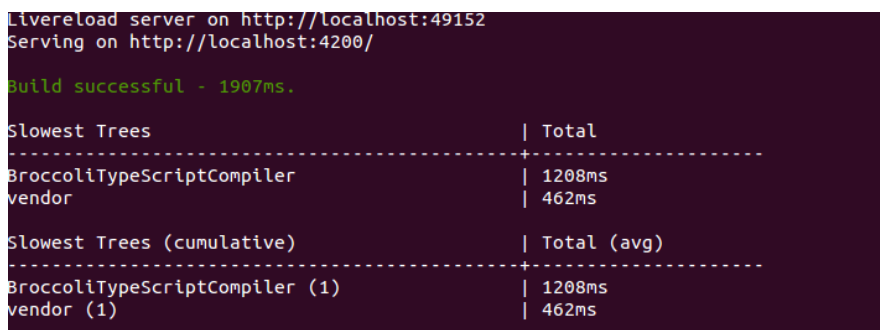


Figura D.2: Consola ejecución. Fuente: <http://codigoxules.org/>.

Si te fijas en la ejecución se llama al comando **ng serve** que será el que utilicemos con Angular CLI para trabajar. Resulta realmente útil al introducir

el último comando nombrado ya que una vez lanzado se detectan automáticamente los cambios a medida que vamos avanzando la página se recarga al guardar el componente que estemos modificando.

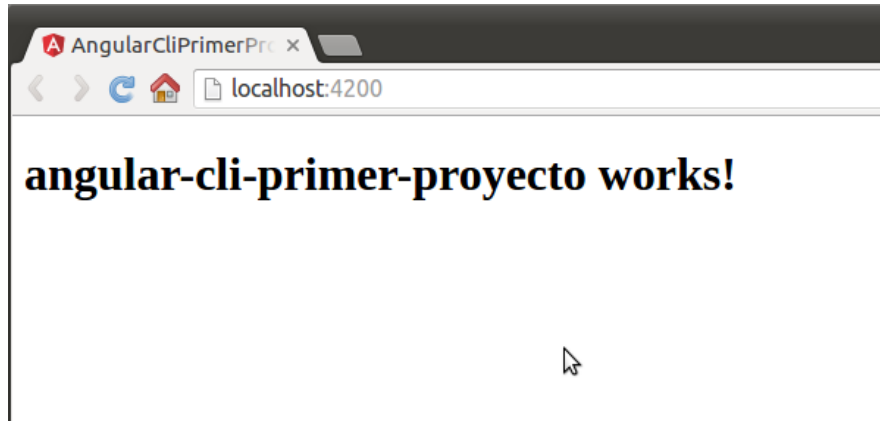


Figura D.3: Resultado después de creado. Fuente: <http://codigoxules.org/>.

Esta es la sencilla manera de crear un proyecto con **Angular CLI**. Después existen una serie de comandos para crear directamente componentes, servicios, clases, interfaces.. etcétera. Algunos de ellos son:

Tipo a crear	Comando
Component	<i>ng component my-new-component</i>
Directive	<i>ng g directive my-new-directive</i>
Pipe	<i>ng g pipe my-new-pipe</i>
Service	<i>ng g service my-new-service</i>
Class	<i>ng g class my-new-class</i>
Interface	<i>ng g interface my-new-interface</i>
Enum	<i>ng g enum my-new-enum</i>

Tabla D.1: Tabla comandos Angular CLI

D.4. Manual del programador

En esta sección hay que tener en cuenta que el autor de este trabajo proyecto a escogido una serie de herramientas, tanto para desplegar la app, como la base de datos como para desarrollar la aplicación pero que de ninguna manera resultan ser ni las únicas ni las mejores simplemente son unas herramientas que ha considerado utilizar pero existen muchas más que no son ni peores ni mejores.

Prerequisitos

1. Instalar Nodejs y MongoDB.
2. Instalar npm
3. Instalar Angular CLI

```
npm i -g @angular/cli
```

4. Descargar el proyecto (*git clone*) o bien desde un soporte
5. Una vez tenemos el proyecto descargado en nuestro sistema local desde el directorio raíz (vía línea de comandos) instalar las dependencias necesarias.

```
npm install
```

Estructura de directorios

La estructura de directorios general es la siguiente:

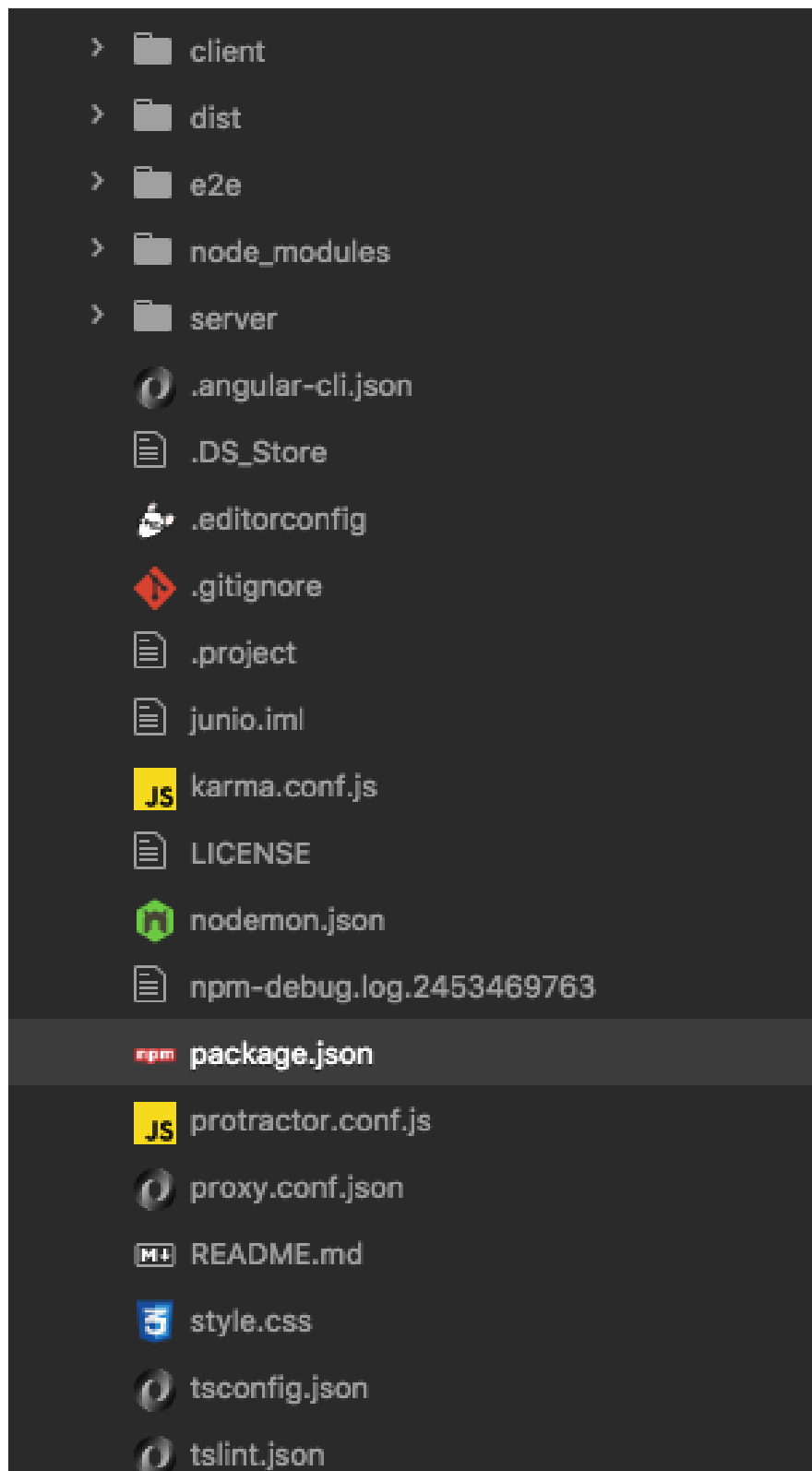
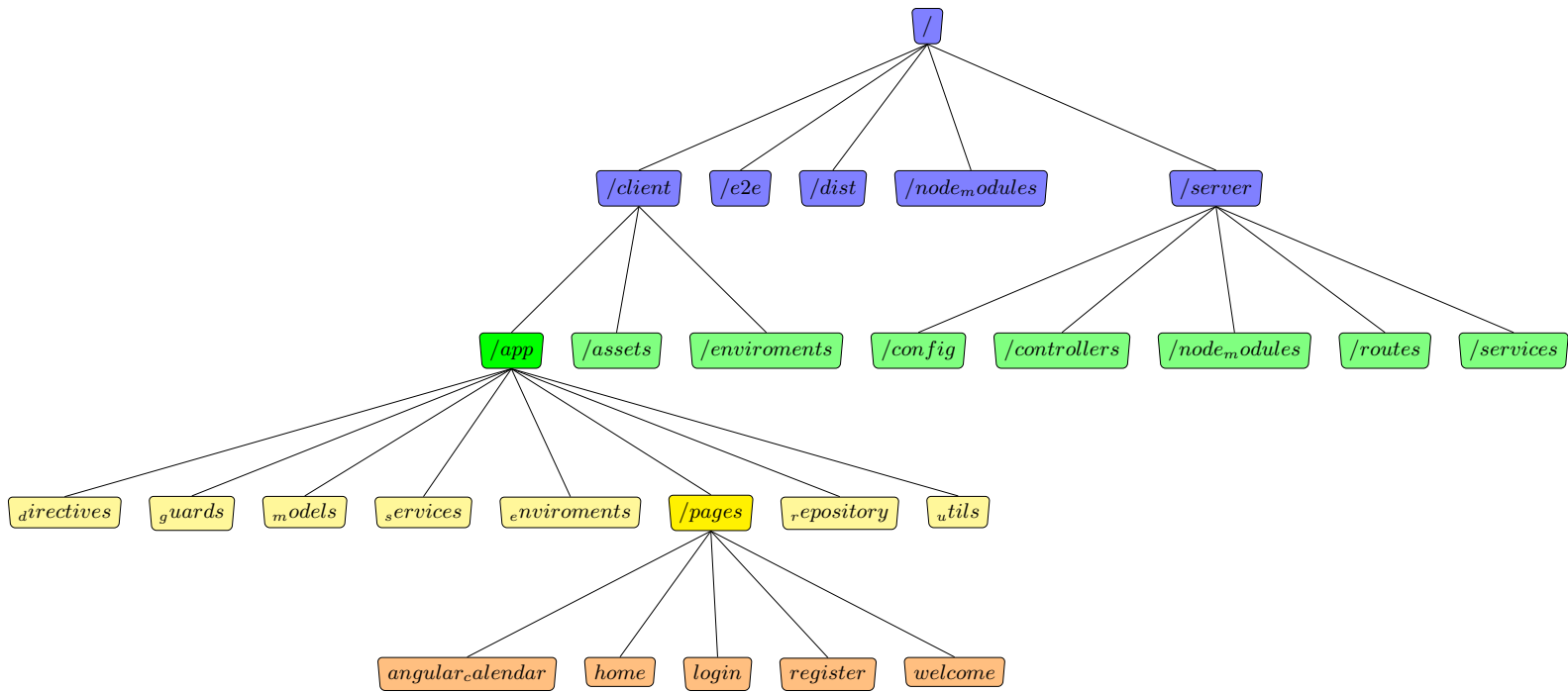


Figura D.4: Vista carpetas general *BarterAPP*. Fuente: Elaboración propia.

- */client/*: Front-End
 - */client/app*: Carpeta App
 - */client/app/directives*: directivas
 - */client/app/guards*: rutas (autenticaciones)
 - */client/app/models*: modelos
 - */client/app/services*: servicios
 - */client/app/enviroments*: entornos
 - */client/app/pages/*: páginas
 - ◇ */client/app/pages/angular_calendar*: calendario
 - ◇ */client/app/pages/home*: home
 - ◇ */client/app/pages/login*: login
 - ◇ */client/app/pages/register*: registro
 - ◇ */client/app/pages/welcome*: vista inicial
 - */client/app/repository*: repositorio (conexión con el servidor)
 - */client/app/utills*: útiles
 - */client/assets*: recursos
 - */client/enviroments*: entornos
- */dist/*: lanzamiento servidor
- */e2e/*: tests
- */node_modules/*: módulos
- */server/*: Back-End
 - */server/config*: configuración
 - */server/controllers*: controladores
 - */node_modules/*: módulos
 - */server/routes*: rutas
 - */server/services*: servicios



APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN³⁴

Observables <http://academia-binaria.com/comunicaciones-http-observables-con-angular2/>

angular2 polling -observables <https://stackoverflow.com/questions/41658162/how-to-do-polling-with-angular-2-observables>

Guards <https://blog.thoughttram.io/angular/2016/07/18/guards-in-angular-2.html>

Interpolacion <http://academia-binaria.com/databinding-el-flujo-de-datos-de-angular2/>

DTO <https://stackoverflow.com/questions/39272947/dto-design-in-typescript-angular2>

ngAfterViewInit <https://angular.io/docs/ts/latest/api/core/index/AfterViewInit-class.html>

PrimeNg

Growl

Login/Register tutorial

AngularCalendar <https://github.com/mattlewis92/angular-bootstrap-calendar>

Multiple components <https://angular.io/docs/ts/latest/tutorial/toh-pt3.html>

Tutorial angular2 components <https://scotch.io/tutorials/creating-your-first-angular-2-components>

<https://desarrolloweb.com/articulos/analisis-carpetas-proyecto-angular2.html>

<https://desarrolloweb.com/articulos/introduccion-componentes-angular2.html>

<http://academia-binaria.com/hola-mundo-en-angular-2/>

Modo desarrollador

Hay que tener en cuenta que tenemos diversas tecnologías dentro del proyecto por lo que resulta un poco costoso tener que ejecutar todas por separado, que es lo que hacía en un principio. Con el modo desarrollador (emula al **ng serve** nombrado antes, tan solo con el comando siguiente ejecutamos MongoDB, Angular, Express y el compilador de Typescript luego ya no nos hace falta nada más y podemos comenzar a trabajar.

```
npm run dev
```

Servidor (Heroku)

Para subir la aplicación al servidor yo he elegido [Heroku](#) y los pasos para lanzar la aplicación son los que se enumeran a continuación:

1. Acceder a la web de *Heroku*, registrarse y crear una nueva app.
2. Instalar *Heroku CLI*
3. Ejecutar los siguientes comandos:

```
heroku login
cd my-project/
git init
heroku git:remote -a your-app-name
```

4. Descargar el proyecto *BarterApp* desde github
5. Editar el servidor(Ruta: *server/config/db.ts*) para introducir un servidor real con MongoDB (lo veremos en la sección siguiente)
6. Ejecutar los siguientes comandos:

```
npm i
ng build -prod or ng build -aot -prod
tsc -p server
git add .
git commit -m "Going to Heroku"
git push heroku master
heroku open
```

7. Una venta se abrirá con tu aplicación ya online

Resumiendo de esta manera utilizamos *Heroku* como si de github se tratase y vamos aplicando los cambios que hacemos a nuestra aplicación al repositorio creado en *Heroku*, la ventaja que ya tenemos la aplicación lanzada en un entorno real. Es una restricción importante que ya no podemos trabajar con una base de datos local sino que tiene que ser un servidor remoto con MongoDB (en este caso) para que la aplicación funcione.

Base de datos (Mlab)

La base de datos escogida es MongoDB, la instalación es fácil y sencilla tan solo hay que ir a la web oficial y descargar la última versión estable disponible. Una vez en nuestro ordenador la ejecutamos y listo. Si tenéis algún problema os recomiendo seguir alguno de los tutoriales que existen en la web. <https://www.adictosaltrabajo.com/tutoriales/mongodb/>

Como sabemos podemos manejar la base de datos desde línea de comandos pero resulta un poco pesado cuando empezamos a tener muchas colecciones o un número de datos alto. Es por eso que yo recomiendo encarecidamente *Robomongo*, la copia de SQL server para bases de datos NoSQL. Interfaz no muy lograda pero perfecta para visualizar nuestras bases de datos e información de una manera más intuitiva.

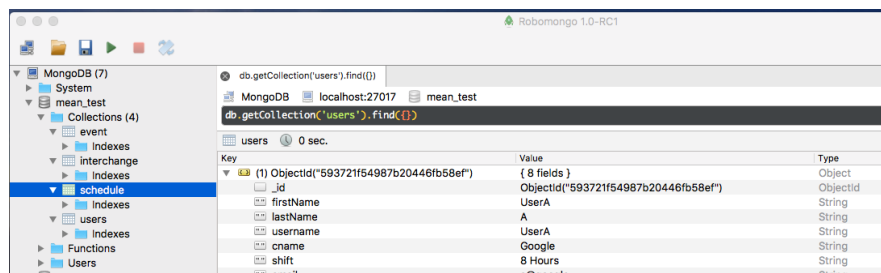


Figura D.5: Vista Robomongo. Fuente: Elaboración propia.

Esta claro que cuando trabajamos con un proyecto en local para realizar pruebas lo mejor es tener una base de datos en local pero cuando queremos algo un poco más serio resulta interesante el tener esa base de datos en un servidor real. Para ello he escogido [Mlab](#), los pasos para crear la base de datos se enumeran a continuación:

1. Acceder a la [Mlab](#), registrarse y crear una cuenta
2. Crear una subscripción a una base de datos (Tiene de pago pero también tiene una versión gratuita que nos vale para realizar pruebas a este nivel)
3. Ejecutar el siguiente comando para establecer la conexión con la base de datos creada

```
mongo ds012345.mlab.com:56789/dbname -u
dbuser -p dbpassword
```

4. Las colecciones las podemos crear directamente desde la propia web, o bien desde la línea de comandos
5. Para probar que la base de datos funciona (El primer comando es para insertar un elemento y el segundo para buscar. En este caso el segundo debería devolver el único elemento insertado.:

```
db.mynewcollection.insert({ "foo" : "bar" })
db.mynewcollection.find()
```

Nota: para empezar a trabajar con este proyecto es necesario crear las tres colecciones de las que ya hemos hablado en secciones anteriores: *Users*, *Events*, *Interchange*, y no hacer nada más puesto que según la configuración de la aplicación se debería añadir los usuarios, eventos y turnos de manera automática cuando el usuario hacer click en los botones adecuados.

■ Stack MEAN

- Mongoose.js (MongoDB): base de datos
- Express.js: backend framework
- Angular 4: frontend framework
- Node.js: entorno tiempo de ejecución
- Otras tecnologías que se utilizan:
 - Angular CLI
 - Bootstrap
 - Font Awesome
 - JSON WEB TOKEN

Documentación de usuario

E.1. Introducción

En este capítulo se detalla como un usuario puede comenzar a usa la aplicación deberemos diferenciar dos aspectos diferentes:

- Aplicación Web.
- Aplicación móvil.

E.2. Requisitos Usuarios

Aplicación Web

Aplicación Móvil

E.3. Primeros pasos

Aplicación Web

Aplicación Móvil