

# **CRONIC**

Aplicación para el control y observación de  
enfermedades de larga duración

**Rubén Sánchez García**

Memoria del Proyecto Final del Grado Multimedia.  
Desarrollo de Aplicaciones Interactivas



Consultor: Kenneth Capseta Nieto

Profesor: Carlos Casado Martínez

Marzo de 2016

# Licencia

- Esta obra está sujeta a una licencia de Reconocimiento No Comercial Sin Obra Derivada 3.0 España de Creative Commons



- El framework Laravel es una aplicación bajo licencia MIT
- Los proyectos de la fundación jQuery están licenciados bajo la licencia especificada en el repositorio del proyecto, o si no se especifica, bajo la licencia MIT.
- Bootstrap se distribuye bajo la licencia MIT. Copyright 2015 Twitter.
- Heroku - Copyright 2000 – 2015 salesforce.com, inc.
- FullCalendar.io - La versión estándar de FullCalendar se lanza bajo una licencia MIT.
- Highcharts - Copyright 2016 Highcharts. Gratuito para uso educativo.
- Select2 - Se distribuye bajo una licencia MIT. Copyright (c) 2012-2015 Kevin Brown, Igor Vaynberg, y los autores que han contribuido.
- Select2 Bootstrap - se distribuye bajo una licencia MIT. Copyright (c) 2012-2015 Florian Kissling
- Bootbox - se distribuye bajo una licencia MIT. Copyright (c) 2011-2014 Nick Payne
- Bootstrap toggle - se distribuye bajo una licencia MIT. Copyright (c) 2011-2014 Min Hur, The New York Times Company
- Datetimepicker - se distribuye bajo una licencia MIT. Copyright (c) 2013 <http://xdsoft.net>
- Moment.js - se distribuye bajo una licencia MIT. Copyright (c) 2011-2016 Tim Wood, Iskren Chernev, y el resto de autores de Moment.js

# Nota del autor

Pese a que la documentación está escrita en castellano, el presente proyecto ha sido realizado en idioma inglés por dos motivos principales:

- La facilidad colaborativa dentro de la comunidad del software libre para un proyecto en el idioma más hablado del mundo
- La conveniencia de utilizar algunas librerías de terceros en ese idioma sin necesidad de configuraciones adicionales

# Abstracto

El registro diario por parte del paciente del estado de una enfermedad de larga duración es un campo poco explorado en el ámbito de las aplicaciones interactivas. Si bien es posible llevar el control utilizando aplicaciones genéricas de toma de notas o incluso un CMS, el disponer de una aplicación específica para ese propósito aporta suficientes ventajas como para ser considerado. Varios ejemplos de dichas ventajas son la posibilidad de añadir avisos específicos para toma de medicinas o visitas a la consulta del profesional sanitario, una interfaz adaptada a nuestro caso de uso o la apertura de un canal de comunicación directo entre el paciente y el profesional sanitario. Es igualmente destacable que la terapia cognitivo-conductual para un amplio abanico de trastornos psicológicos incluye como parte imprescindible esta clase de registros <sup>1</sup>.

La aplicación web CRONIC nace con la intención de proponer una solución que cubra las necesidades de esa problemática. Mediante la creación de un perfil de usuario, el paciente deberá ser capaz de guardar un pequeño texto con la descripción de su estado, añadir un historial de medicinas aplicables a su caso y registrar eventos de interés como futuras visitas a la consulta de su especialista. Asimismo el profesional sanitario podrá crear un perfil de usuario que quedará asociado a sus pacientes y permitirá la recepción en su pantalla principal de las entradas de registro que estos hayan decidido enviarle.

**Palabras clave**— memoria, tfg, cronic, registro enfermedad, comunicación doctor paciente, terapia cognitivo conductual

---

<sup>1</sup>Dubord 2011

# Abstract

The daily journaling by the patient of a long duration illness is a field which has only been marginally explored in the field of interactive applications. Even though it's possible to log it by using generic note taking applications or even a CMS, the fact of having a specific application for that purpose gives enough benefits so to make it worth. A few examples are the possibility of adding tailored reminders for the taking of medicines or medical appointments, an interface specifically tailored to our use case or the opening of a communication channel between the patient and the medical professional. It is also worth to mention that the cognitive-conductual therapy for a wide range of psychological disorders includes this kind of logging as an essential part. <sup>2</sup>.

The web application CRONIC is born with the aim of propose a solution that covers the necessities of this problematic. By creating a user profile, the patient will be able to keep a brief description of his or her daily state, add a history of medicines that apply to the case and include interesting events such as future medical appointments. The medical professional will also be able to create a user profile associated to the patient, which will enable the retrieval of entries that the patient might have considered worth to send for review.

**Keywords**— end of degree project paper, illness monitoring, cronic, doctor patient communication, cognitive conductual therapy

---

<sup>2</sup>Dubord 2011

# Índice general

<b>1. Introducción</b>	<b>11</b>
1.1. Motivación . . . . .	11
1.2. Propuesta . . . . .	12
<b>2. Descripción</b>	<b>13</b>
2.1. Control de la enfermedad . . . . .	13
2.2. La aplicación web como plataforma ideal . . . . .	14
2.3. CRONIC . . . . .	15
<b>3. Objetivos</b>	<b>16</b>
3.1. Objetivos generales . . . . .	16
3.2. Objetivos específicos . . . . .	16
3.3. Objetivos a largo plazo . . . . .	17
<b>4. Marco Teórico</b>	<b>18</b>
4.1. Estudios previos . . . . .	18
4.2. Herramientas actuales . . . . .	18
<b>5. Contenido</b>	<b>19</b>
5.1. Contenidos de la aplicación web . . . . .	19
5.2. Otros contenidos . . . . .	22
<b>6. Metodología</b>	<b>24</b>
6.1. Enfoque metodológico . . . . .	24
6.2. Metodología de programación y gestión del proyecto . . . . .	25
<b>7. Arquitectura de la aplicación</b>	<b>28</b>
<b>8. Plataforma de desarrollo</b>	<b>29</b>
<b>9. Planificación</b>	<b>32</b>
<b>10. Proceso de Trabajo</b>	<b>36</b>

---

<b>11.Prototipos</b>	<b>45</b>
<b>12.Perfiles de Usuario</b>	<b>52</b>
12.1. Personajes de usuario . . . . .	52
<b>13.Usabilidad</b>	<b>54</b>
13.1. Partes del test . . . . .	54
<b>14.Seguridad</b>	<b>57</b>
<b>15.Versiones</b>	<b>58</b>
<b>16.Instrucciones</b>	<b>59</b>
<b>17.Bugs conocidos</b>	<b>61</b>
<b>18.Proyección de Futuro</b>	<b>62</b>
<b>19.Presupuesto</b>	<b>63</b>
<b>20.Análisis de mercado</b>	<b>64</b>
20.1. Herramientas actuales . . . . .	64
<b>21.Conclusiones</b>	<b>66</b>
21.1. Conclusiones sobre temas teóricos . . . . .	66
21.2. Conclusiones sobre temas prácticos . . . . .	66
<b>A. Lista de entregables</b>	<b>68</b>
A.1. Documentos . . . . .	68
<b>B. Código fuente</b>	<b>69</b>
<b>C. Capturas de pantalla</b>	<b>70</b>
<b>D. Librerías utilizadas</b>	<b>79</b>
D.1. Librerías para PHP . . . . .	79
D.2. Librerías para Javascript . . . . .	79
D.3. Librerías para CSS . . . . .	80
<b>E. Guía de estilo</b>	<b>81</b>

# Índice de figuras

5.1. Platform as a service. . . . .	20
6.1. Funcionamiento del método Scrum (elaboración propia). . . . .	26
6.2. Vista de las historias de usuario. . . . .	27
6.3. Vista del listado de errores. . . . .	27
7.1. Arquitectura de la aplicación. . . . .	28
8.1. El software Vagrant nos permite ejecutar un entorno Ubuntu precon- figurado para desarrollo desde OSX. . . . .	30
9.1. Listado de tareas en el diagrama . . . . .	34
9.2. Representación gráfica del proceso . . . . .	35
10.1. Interfaz de NinjaMock . . . . .	38
10.2. Vista de la pizarra única de Kanban. . . . .	41
11.1. Algunos bocetos preliminares anteriores a la realización de los proto- tipos. . . . .	46
11.2. Protitipo de la pantalla de introducción del diario . . . . .	47
11.3. Protitipo de la pantalla de introducción de recordatorios. La opción de email no ha sido incorporada en la versión final . . . . .	48
11.4. Protitipo de la pantalla de vista del diario . . . . .	49
11.5. Protitipo de la pantalla de editar perfil . . . . .	50
11.6. Protitipo de la pantalla de búsqueda, no incluida en esta versión. . . . .	51
16.1. Ejemplo de Impromptu . . . . .	60
16.2. Ejemplo de Chardin . . . . .	60
C.1. Portada. . . . .	70
C.2. Formulario de registro . . . . .	71
C.3. Entrada. . . . .	71
C.4. Panel de control. . . . .	72
C.5. Introducción de diario. . . . .	72



---

C.6. Introducción de eventos. . . . .	73
C.7. Listado de entradas. . . . .	73
C.8. Gráfico de dolor. . . . .	74
C.9. Vista de calendario. . . . .	74
C.10. Edición de perfil. . . . .	75
C.11. Perfil actualizado. . . . .	75
C.12. Distribución de menú en modo tablet. . . . .	76
C.13. Distribución de menú en modo teléfono. . . . .	77
C.14. Menú desplegado en modo teléfono. . . . .	78

# Índice de cuadros

8.1. Herramientas utilizadas. . . . .	29
8.2. Lenguajes y Librerías. . . . .	29
19.1. Presupuesto. . . . .	63

# Capítulo 1

## Introducción

### 1.1. Motivación

Durante los últimos años se ha popularizado el fenómeno de registrar todos los datos, momentos, ideas y conceptos de nuestra vida. Con la ayuda de las tecnologías móviles que nos permiten ejecutar aplicaciones en cualquier lugar, hemos visto la llegada de herramientas para registrar y catalogar lo que comemos, las horas que dormimos, cuantas tazas de café bebemos al día, los pasos dados y nuestras sesiones en el gimnasio, qué tareas tenemos pendientes, a qué personas nos han presentado hoy e incluso cuanto han ejercitado nuestras mascotas.

Sin embargo las aplicaciones de la rama sanitaria suelen ser de un tono amable, enfocándose a registrar nuestra actividad, indicando que a cuanta más actividad, más salud. También acostumbran a incluir apartados para registrar alergias o medicinas, pero nunca da la impresión de que estas aplicaciones se dirigen a la persona realmente enferma, más bien a alguien que simplemente se preocupa por su salud.

Así pues, el panorama de las aplicaciones dedicadas al registro diario de incidencias en lo que respecta a enfermedades serias de larga duración quedaba limitado a un pequeño puñado de ejemplos en las tiendas de aplicaciones para iOS y Android. Un gran porcentaje son de pago y muchas de las que se ofrecen gratuitamente son de una calidad deficiente, particularmente en lo que respecta a la interfaz y usabilidad.

Considerando entonces que este tipo de programas ha recibido muy poca atención por parte de los desarrolladores, tenemos ante nosotros una oportunidad para crear una aplicación para las personas con enfermedades serias que pueda aportar funcionalidades interesantes como una comunicación doctor paciente u otros añadidos de interés.

## 1.2. Propuesta

A lo largo del presente trabajo se va a desarrollar una aplicación web responsiva con la intención de aportar una herramienta de utilidad al campo de las aplicaciones sanitarias. Con nuestro desarrollo vamos a posibilitar que el paciente pueda registrar los eventos e incidencias de su enfermedad, que pueda establecer un vínculo comunicativo con el profesional sanitario que le atiende e incluso registrar su toma de medicamentos o programar las próximas visitas a la consulta mediante un calendario.

# Capítulo 2

## Descripción

### 2.1. Control de la enfermedad

Cuando una enfermedad se cronifica o simplemente evoluciona durante un periodo de tiempo medio o largo, llevar un registro de su estado es de suma importancia. Por desgracia, la única circunstancia en la cual el registro puede ser llevado diariamente por un profesional es cuando el paciente se encuentra ingresado en un centro hospitalario. Para el resto de situaciones, la observación del estado se realiza únicamente con la frecuencia con la cual se programen las visitas al facultativo.

En esos casos, la observación diaria de la evolución de los síntomas quedará bajo responsabilidad del paciente. Salvo en las ocasiones concretas en las cuales a este le recomienden llevar un registro, habitualmente se limitará a usar la memoria e intentar recordar durante qué días se sintió mejor o peor, lo cual no es extremadamente fiable en lo que respecta a proponer un diagnóstico y tratamiento. La inclusión en la rutina diaria de un control y registro mediante una herramienta interactiva no solo aporta los beneficios habituales de un registro normal y corriente como el que pudiera hacerse con papel y bolígrafo, sino que añade una serie de ventajas inherentes a la plataforma utilizada.

#### 2.1.1. Beneficios para el paciente

Uno de los beneficios más interesantes que el paciente percibirá llevando registros es la observación de patrones. Los síntomas nuevos o agravados que sin registros pudieran parecer arbitrarios, con un estricto control diario podrán relacionarse a otros factores como la toma de ciertas medicinas, ingesta de uno u otro alimento, ejercicio físico o falta de este por citar algunos ejemplos. De este modo es posible descubrir factores que contribuyan a mejorar la salud del paciente.

Igualmente existe la posibilidad de que en cierto momento el tratamiento o simplemente el estilo de vida se degrade, con lo cual darse cuenta a través de los registros

puede desencadenar una pronta corrección.

Existen diversos estudios que relacionan el registro diario con la mejora intrínseca del estado de salud (Purcell 2006) Si bien los estudios no son conclusivos y ni siquiera se refieren exclusivamente a la escritura de datos de salud sino al llevar un diario genérico sobre cualquier temática, se puede deducir de ello que en nuestro caso también debería existir una sensible mejora del estado de ánimo.

### **2.1.2. Aplicaciones en diversas terapias**

En el apartado anterior se mencionan algunos factores que benefician al paciente en lo que respecta a controlar enfermedades eminentemente físicas. Una de las facetas más interesantes del llevar un diario médico es que el solo hecho de llevarlo ya es una parte indispensable en la terapia cognitivo conductual aplicada a una serie de trastornos psicológicos. No es el propósito de este estudio el entrar en detalles exhaustivos sobre estos beneficios, pero como pequeño resumen se puede mencionar que el paciente notará mejora en lo que respecta a gestionar la ansiedad y la depresión, reducir el estrés o incluso priorizar problemas y miedos para entenderlos mejor.

### **2.1.3. Beneficios para el profesional sanitario**

El principal beneficio para el profesional sanitario es que al aportar el paciente un registro fidedigno de sus estados, este puede tomar decisiones en lo que respecta al diagnóstico y tratamiento con una mayor fiabilidad que en el supuesto caso de basarse en recuerdos que pueden no ser exactos.

La inclusión de materiales multimedia como fotografías, complementa el registro de texto y facilita la comprensión de lo que en este se detalla. Puede ser particularmente de utilidad para enfermedades muy visibles como heridas de gran tamaño, úlceras o algunos tipos de tumores.

Por último es necesario reseñar que el formato de registro mediante una aplicación interactiva abre una serie de nuevas posibilidades en lo que respecta al canal de comunicación entre el paciente y el profesional sanitario. Esto puede traducirse en una mejora del servicio que en algunos contextos de la sanidad privada podría utilizarse para cobrar un importe extra.

## **2.2. La aplicación web como plataforma ideal**

Dentro del abanico de posibilidades a la hora de crear una aplicación interactiva, la opción que mejor se ajusta a las necesidades del proyecto es una aplicación web.

Los avances en CSS para realizar aplicaciones que adapten su medida de pantalla al dispositivo que las utilice sumado a la proliferación de conexiones a internet en dispositivos móviles, que en 2016 llegó a 3.790 billones de usuarios<sup>1</sup>, hacen que el hecho de acceder a una web móvil desde cualquier lugar no represente un problema. Al mismo tiempo existen otras ventajas como la compatibilidad en múltiples dispositivos. De hecho, cualquier aparato que disponga de un navegador web debería ser capaz de visualizar la aplicación.

Existen algunas desventajas. Por ejemplo, no es posible aprovechar las API del teléfono para funciones como utilizar la cámara o enviar notificaciones push. Para conseguir esto debería utilizarse una aplicación instalable, lo cual nos aportaría ventajas adicionales pero también una serie de problemas como por ejemplo el incremento del tiempo necesario para desarrollo.

## 2.3. CRONIC

CRONIC es el nombre de la aplicación desarrollada para solucionar la problemática expuesta. Por el momento se ofrece únicamente en formato aplicación web, aunque no se descartan versiones adicionales para móviles en un futuro. Ha sido programada en *PHP* usando el *framework Laravel* para el *backend* y en *javascript* con el *framework jQuery* para el *frontend*.

Su funcionalidad principal permite dar de alta un usuario y asociar datos de una enfermedad. También es posible añadir el nombre del doctor especialista. El paciente registrará su estado con la frecuencia deseada. Si así lo desea, podrá marcar algunas entradas como especialmente interesantes. Como funcionalidad adicional será posible añadir avisos para la toma de medicinas y futuras consultas médicas, siendo posible consultar estas últimas en un calendario.

Por otra parte, el doctor se habrá dado de alta con un perfil de profesional. En su pantalla principal podrá acceder al flujo de registros del paciente, o bien completo o bien filtrando por las entradas que este haya marcado como interesantes.

---

<sup>1</sup><http://www.slideshare.net/wearesocialsg/digital-in-2016>

# Capítulo 3

## Objetivos

### 3.1. Objetivos generales

- Desarrollar una aplicación web accesible desde dispositivos móviles (diseño responsivo) para llevar el control de enfermedades de larga duración
- Utilizar la experiencia del desarrollo para aprender nuevas tecnologías como por ejemplo el *framework Laravel*
- Implementar la base tecnológica de la aplicación (el *backend*) de una manera que haga factible el abrir la web al público general una vez finalizado el TFG con el posible crecimiento que esto conllevaría
- Conseguir un producto final que sea de utilidad para los tipos de usuario a los que se dirige

### 3.2. Objetivos específicos

- La aplicación deberá permitir el registro del estado del paciente diario o con la frecuencia que este desee.
- Deberá existir un apartado para especificar las medicinas tomadas y visitas al médico.
- Las medicinas se gestionarán mediante recordatorios. En la versión web el recordatorio se limitará a enviar un email a la dirección incluida en el perfil del paciente
- Las futuras visitas a la consulta se gestionarán mediante un calendario



- La aplicación deberá permitir el marcar entradas para el envío al facultativo responsable del tratamiento. Estas serán vistas por este al entrar en su sesión de usuario

### 3.3. Objetivos a largo plazo

Estos objetivos no forman parte del bloque de trabajo presentado en este TFG, pero sí del bloque de trabajo futuro que se realizará a partir de la entrega final.

- Valorar la conveniencia de migrar a una base de datos *NoSQL* para facilitar la comunicación con aplicaciones móviles
- Como alternativa, explorar la posibilidad de crear el propio *feed JSON* desde el propio *Laravel*, o con herramientas que lo faciliten, como *Lumen*<sup>1</sup>
- Adaptar el código a una aplicación móvil realizada con *PhoneGap* para así aprovechar al máximo las API de los dispositivos móviles
- Trabajar en la difusión del producto y el crecimiento de usuarios
- Estudiar la viabilidad de establecer la funcionalidad de registro como gratuita pero la comunicación con el doctor como función de pago
- Estudiar nuevas funciones sociales, como foros de comunicación entre pacientes de la misma enfermedad

---

<sup>1</sup><https://lumen.laravel.com/>

# Capítulo 4

## Marco Teórico

### 4.1. Estudios previos

En lo que respecta a la literatura analizando los beneficios de la escritura a modo de diario para registrar la evolución de la enfermedad, encontramos muy pocos ejemplos. Algunos estudios prueban los beneficios de tal actividad (Shetzer 2007) en contextos genéricos de enfermedad o edad avanzada. Sin embargo hay algunas enfermedades que han merecido especial atención en ese ámbito. El cáncer es una de ellas, existiendo gran variedad de estudios sobre los beneficios de llevar un registro diario (Smith y col. 2005).

Si bien es cierto que estos textos se están refiriendo a un diario en el sentido más común del término (de ahí el uso del término inglés *journaling*), no existe gran cantidad de literatura sobre el registro de los conceptos médicos puros abstraídos del bloque principal de texto. A falta de estudios que prueben su beneficio, solo podemos sacar conclusiones del material existente e intentar extrapolar los beneficios conocidos al contexto de nuestro producto.

Un campo donde sí existe abundante documentación es la aplicación de este tipo de registros en las terapias cognitivo-conductuales y el llamado auto-registro. Llamado también registro de pensamiento, se considera una de las herramientas principales para el retorno del paciente a la racionalidad (Dubord 2011). Es interesante destacar cómo algunos estudios sobre dichas terapias destacan también los beneficios en la enfermedad crónica a un nivel más genérico (Taylor 2006).

### 4.2. Herramientas actuales

Esta temática se mencionará más adelante en el capítulo 20, Análisis de mercado.

# Capítulo 5

## Contenido

### 5.1. Contenidos de la aplicación web

La aplicación web se encuentra alojada en <http://cronic.herokuapp.com/>. *Heroku* es un proveedor de *PaaS*<sup>1</sup>, lo que significa que los pasos asociados a la configuración de la plataforma de alojamiento no serán necesarios. Si en un alojamiento al uso se espera que configuremos la base de datos, los permisos de usuarios y la seguridad, en una *PaaS* únicamente se espera que subamos el código de nuestra aplicación, quedando la configuración a cargo del proveedor del servicio. El siguiente gráfico ilustra las diferencias entre uno y otro modo:

---

<sup>1</sup><https://www.salesforce.com/paas/overview/>

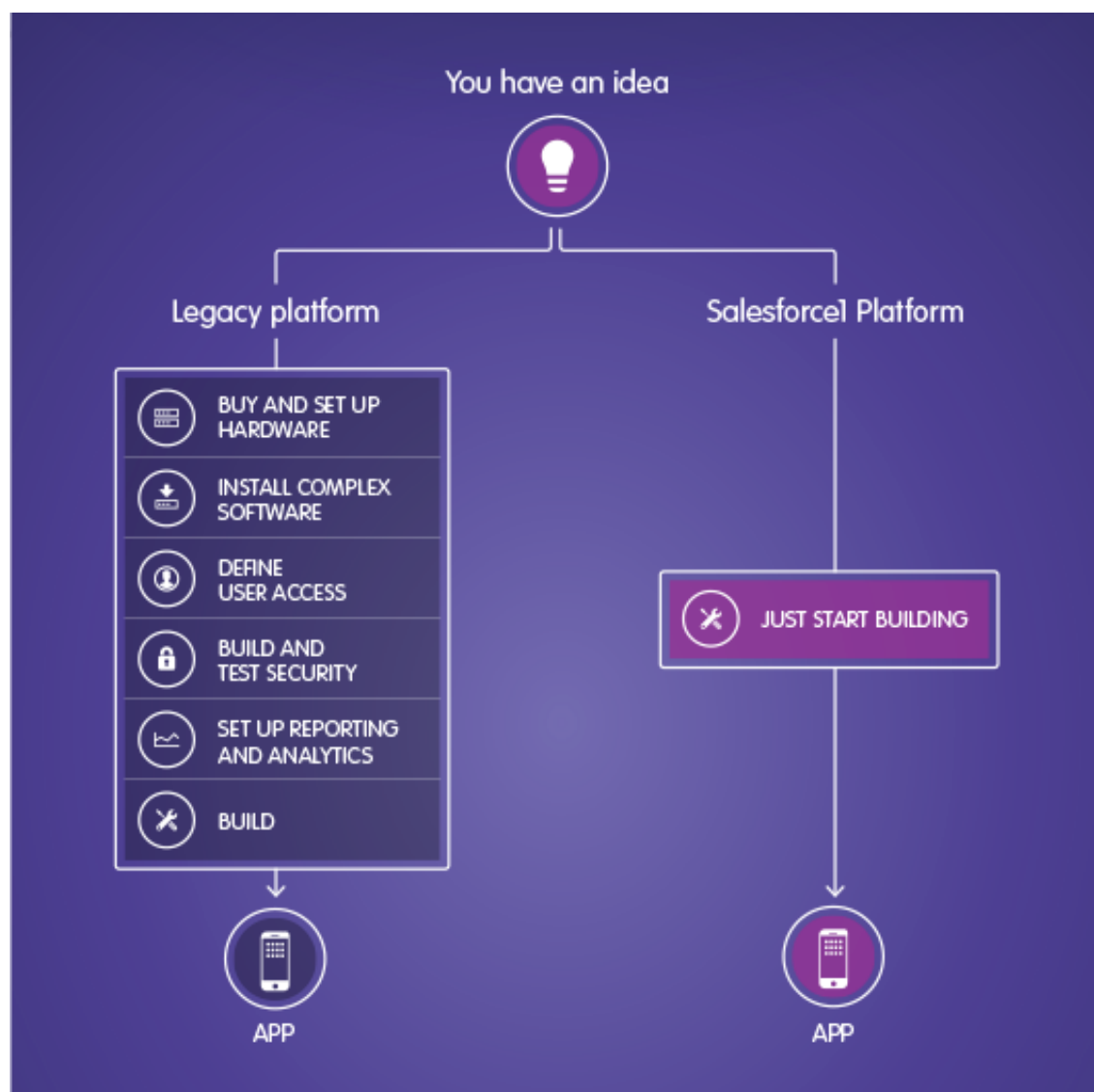


Figura 5.1: Platform as a service.

### 5.1.1. Registro de usuarios

La primera vez que el usuario accede a nuestra aplicación, es necesario registrarse para obtener unas credenciales. Actualmente solo es posible el registro en la base de datos propia. Existe una anotación en el gestor de *Scrum*<sup>2</sup> (Taiga) para implementar el registro vía redes sociales<sup>3</sup>, pero muy probablemente se incluya en los *sprint* posteriores a la entrega final de este trabajo. Se expondrán más detalles sobre *Agile/Scrum* y Taiga en futuros capítulos.

Para registrarse únicamente se solicita el nombre, apellidos y una dirección de email. Adicionalmente se requiere saber si la persona se registra como paciente o como doctor, ya que de ello dependerá la funcionalidad disponible en cada caso. El

<sup>2</sup><https://www.scrum.org/Resources/What-is-Scrum>

<sup>3</sup><https://tree.taiga.io/project/rubenwap-cronic/us/15>

---

resto de datos se aportarán en la sección de configuración.

### 5.1.2. Panel de control

El panel de control es la vista principal que el usuario recibirá al introducir sus credenciales en la aplicación. En una primera versión, se incluirán las siguientes partes:

- Vista destacada de la última entrada guardada. Puede valorarse aumentar el número e incluir dos o tres entradas. La vista incluirá el título, el icono para representar el estado de salud del paciente y un extracto del cuerpo principal del texto.
- Vista preliminar o versión reducida del gráfico del apartado *Progreso*, en el cual se representan los picos del estado de salud del paciente, utilizando para ello la escala que ha de seleccionarse en cada entrada del diario que se registre.
- Lista de próximos recordatorios de calendario, caso de haberse registrado alguno en la página correspondiente.

### 5.1.3. Vista de todos los registros

Una vista en formato tabla de todas las entradas con paginación. Se implementarán una serie de filtros para poder seleccionar un sub grupo de resultados.

### 5.1.4. Vista de introducción de registros

El formulario principal para la introducción de registros. Se compone de las siguientes partes:

- Sensación: Basado en una escala de medición del dolor a través de caras (Bieri y col. 1990). No existe una única escala sino varias, cada una con distintos matices. El desafío para un producto multimedia consiste en simplificar la escala (a menudo la cantidad de caras es excesiva) y depurar el estilo gráfico para que sea atractivo al tiempo que comprensible. La meta es que cada cara tenga matices distintos que la hagan reconocible.

En este formulario se presentan en un carrusel. El usuario deberá situar el carrusel en la posición correspondiente a su elección de cara.

- Título: Un título genérico explicando el tema principal de la entrada.
- Cuerpo: Detalle expandido del tema.

En el futuro está previsto implementar la funcionalidad de incorporar datos estructurados. Es decir, será posible añadir campos dinámicamente para no limitarse a escribir texto sino datos como presión sanguínea, pulsaciones, nivel de azúcar u otras variables a elección del paciente.

### 5.1.5. Vista de eventos / calendario

Se incorporará un calendario en el cual podrán añadirse eventos. El formulario para añadirlos tiene en cuenta conceptos habituales para los pacientes como por ejemplo citas en la consulta médica o comienzo de la toma de un medicamento.

### 5.1.6. Vista de progreso

Cuando el paciente selecciona una cara de la escala de dolor en el formulario principal, se asigna un valor numérico a esa cara. De ese modo, es posible llevar un control gráfico del progreso de la enfermedad en términos generales. La primera versión de la aplicación permitirá ver la representación de la última semana. En el futuro se añadirán opciones adicionales para personalizar completamente el gráfico.

### 5.1.7. Vista de configuración

En esta vista es posible completar el perfil del paciente de manera totalmente opcional con datos relativos a su enfermedad. Si así lo desea también será posible seleccionar un doctor (de entre los cuales se hayan dado de alta en la aplicación) para poder enviar anotaciones a este en un *feed*.

## 5.2. Otros contenidos

La siguiente lista de elementos no se incluirá en la versión presentada como objeto de este trabajo, pero a modo informativo, las siguientes características serán implementadas en el futuro:

- Creación de foros temáticos sobre enfermedades. El usuario podrá activar su participación en ellos de manera opcional
- Creación de un chat en tiempo real con los usuarios registrados
- Implementación de RESTful APIs desde nuestra base de datos en PostgreSQL con la intención de hacerlos legibles desde una aplicación móvil
- Valorar la creación de una aplicación móvil nativa, inicialmente en un *framework* multiplataforma como *PhoneGap*.

### 5.2.1. Página web

La página web del producto actualmente no cumple ninguna otra función excepto la de mostrar una breve presentación de sus características y permitir el registro del usuario. En el futuro se ampliarán las funcionalidades. Se difundirán las aplicaciones nativas y se incluirá información corporativa si se estima necesario constituirse como empresa para facilitar los trámites relacionados a la gestión de datos personales y otra información privada del usuario.

Como alternativa a la comercialización del producto se propone liberarlo como software libre con licencia no comercial.

# Capítulo 6

## Metodología

### 6.1. Enfoque metodológico

A la hora de afrontar la metodología del siguiente trabajo, se han tenido en cuenta una serie de puntos de interés, organizados en categorías según su importancia. Dependiendo de si esta es alta o baja, se les otorga más o menos tiempo de investigación y desarrollo. A continuación se exponen los pasos ordenados según su importancia:

#### 6.1.1. Muy importante

- Documentarse sobre los beneficios del registro en la salud el usuario para poder planear una aplicación coherente y de utilidad
- Estudiar la documentación de *Laravel* para poder realizar la base de la aplicación de manera ágil con el menor número de problemas posibles
- Estudiar conceptos avanzados de bases de datos para poder diseñar la de la aplicación de manera lógica

#### 6.1.2. Moderadamente importante

- Revisar los principios de diseño y usabilidad que contribuyan a una mejor experiencia de usuario
- Investigar distintos *plugins* de *jQuery* que puedan contribuir a facilitar la interacción con la interfaz
- Estudiar la idoneidad de *Heroku* como plataforma para lanzar la aplicación



### 6.1.3. Poco importante

- Trazar un plan de viabilidad de la aplicación como producto comercial

Una vez estudiados estos puntos, se han integrado o descartado para la planificación definitiva. Este plan, expuesto en el Capítulo 9, muestra la lista maestra de las tareas llevadas a cabo en el presente trabajo y el orden aplicado a estas.

## 6.2. Metodología de programación y gestión del proyecto

Si el apartado anterior se refería a consideraciones generales y el futuro capítulo 9 explicará lo relativo a la gestión del proyecto, el capítulo actual debe detallar la técnica que se ha utilizado para llevar a buen término la programación de la aplicación. Se trata de *Scrum*, una metodología basada en el movimiento *Agile* muy popular en la actualidad, que pese a estar pensada para equipos, también puede aplicarse al trabajo de una sola persona, como el caso que nos ocupa.

### 6.2.1. Algunos principios de Agile relevantes a nuestro caso (Beck y col. 2001)

- Los cambios de requisitos son bienvenidos incluso a mitad de desarrollo
- Las iteraciones del software funcionando se van entregando con rapidez
- El progreso se mide según si el software funciona o no
- La prioridad es para el producto funcional, no para la documentación detallada

Por supuesto es necesario dejar fuera los principios que hacen hincapié en la colaboración entre el equipo. Se requiere un estudio separado para detallar las características y beneficios de esta metodología, pero como resumen cabe destacar la importancia que se le da a conseguir piezas funcionando aunque no sean perfectas a la primera. En el caso actual no es posible organizar las reuniones *Scrum* diarias, ya que con una persona no tienen sentido, pero sí podemos trabajar con un *Backlog*, o lista de tareas (o historias) pendientes.

De esta lista vamos seleccionando elementos y ubicándolos en un *Sprint* regular, para forzarnos a una cierta productividad. Cuando se trabaja en equipo esta etapa suele durar entre una y dos semanas. En un hipotético caso de dos semanas, la semana en la cual comienza un *Sprint*, primero se repasa y analiza cómo ha funcionado

el de la semana anterior. La segunda semana, se pueden filtrar y elegir elementos del *Backlog* para comenzar de nuevo la siguiente semana.

En el gráfico adjunto puede entenderse esta idea. Se ha utilizado como ejemplo un proyecto con tres versiones: *Integration*, para uso por los desarrolladores, *Staging*, para uso por los *testers* y *Production*, para uso general por todos los públicos. El éxito de un *Sprint* provoca el traslado del código de un entorno a otro.

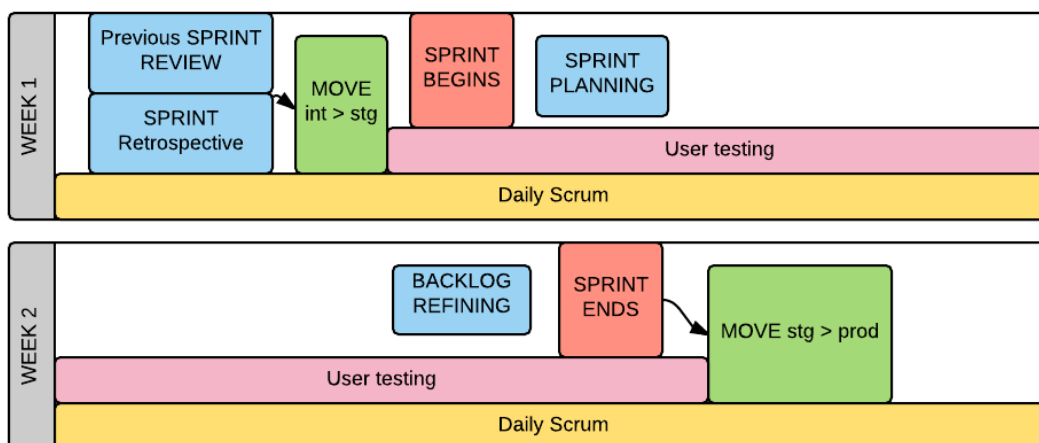


Figura 6.1: Funcionamiento del método Scrum (elaboración propia).

### 6.2.2. Gestión de *Agile/Scrum* mediante el programa *Taiga*

A continuación se muestran varias capturas de pantalla para los listados de mejoras y errores de la aplicación dentro del gestor *Taiga*.

*Taiga* no es un programa tan poderoso y versátil como JIRA<sup>1</sup>, uno de los más conocidos en su campo, pero tiene la ventaja de ser software libre, con lo cual está al alcance de cualquiera.

### 6.2.3. Metodología Kanban

Antes de la entrega final se decide cambiar la metodología para llevar el control de las partes pendientes. Si bien *Agile* funciona, se considera que el trabajo de controlar los sprints, actualizar las pizarras y demás, tiene sentido dentro del contexto de un equipo. Al ser este el trabajo de una sola persona, se cambia al método *Kanban*, mucho más apropiado y del cual se darán más detalles en el apartado del proceso de trabajo en el capítulo 10.

<sup>1</sup><https://www.atlassian.com/software/jira>

CRONIC BACKLOG

0% 335 defined points 0 closed points 0 points / sprint

CUSTOMIZE YOUR BACKLOG GRAPH  
To have a nice graph that helps you follow the evolution of the project you have to set up the points and sprints through the **Admin**

SHOW FILTERS SHOW TAGS + ADD A NEW USER STORY

Votes	User Stories	Status	Points
0	#1 Replace highcharts placeholder with Laracharts	New	19
0	#2 Calendar events page	New	19
0	#3 User settings page	New	29
0	#4 Filter "all entries" results	New	31
0	#5 Edit and Delete from article page	New	15
0	#7 Find and apply bootstrap theme	New	30
0	#8 Create versions of pages for doctor condition	New	38

1 SPRINTS  
Short sprint up to PAC2  
28 Mar 2016-06 Apr 2016 0 closed 40 total

#13 Watch Laracasts videos 40

SPRINT TASKBOARD

Figura 6.2: Vista de las historias de usuario.

CRONIC ISSUES

+ NEW ISSUE

Type	Severity	Priority	Votes	Subject	Status	Created	Assigned to
●	●	●	0	#16 Save new article doesn't ...	New	29 Mar 2016 22:25	Ruben Sa...
●	●	●	0	#14 Add migration to articles..	Postponed	28 Mar 2016 21:20	Ruben Sa...
●	●	●	0	#6 Article page does not sho...	New	28 Mar 2016 20:53	Ruben Sa...

FILTERS

Subject or ref

TYPE

STATUS

SEVERITY

PRIORITIES

TAGS

ASSIGNED TO

CREATED BY

Figura 6.3: Vista del listado de errores.

# Capítulo 7

## Arquitectura de la aplicación

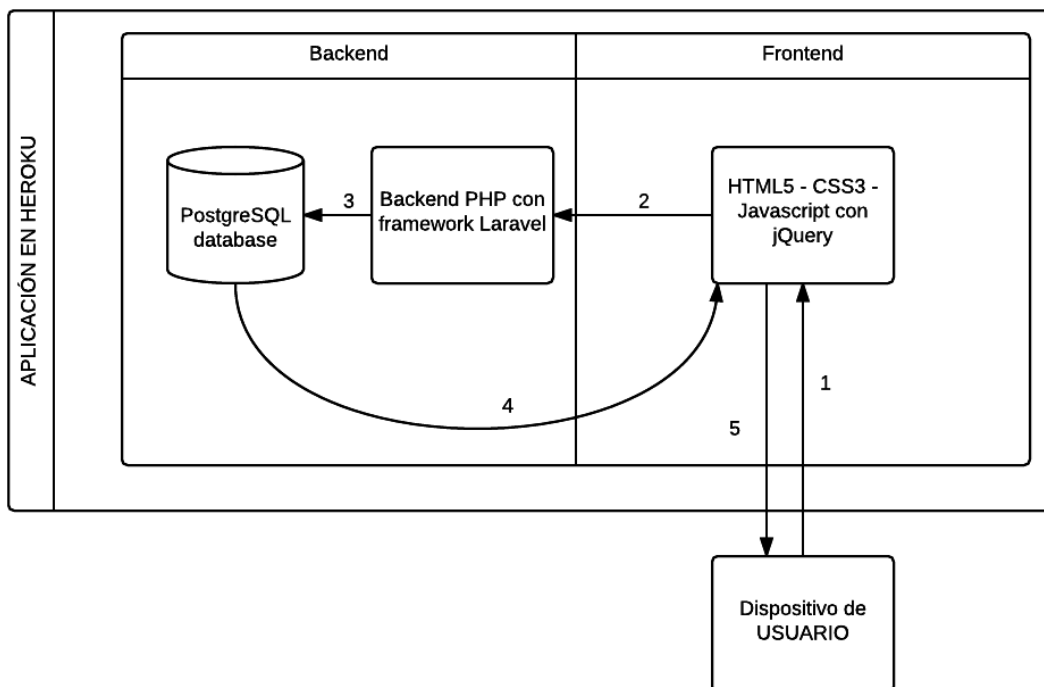


Figura 7.1: Arquitectura de la aplicación.

En una aplicación que utilice el paradigma de diseño MVC (Modelo Vista Controlador)<sup>1</sup>, el usuario accede a la vista en la cual formula sus peticiones (en este caso, las pantallas de la interfaz basadas en HTML, CSS, JS). La petición se envía a procesar al Controlador (la lógica de la aplicación, en este caso programada en PHP). De ser necesario, se realizarán peticiones al modelo de datos (almacenado aquí en una base de datos PostgreSQL). El controlador recibirá la respuesta del modelo de datos, y lo enviará a la vista, que es lo que recibirá el usuario.

<sup>1</sup><https://es.wikipedia.org/wiki/Modelo-vista-controlador>

# Capítulo 8

## Plataforma de desarrollo

Hardware	Software	Web apps
Mac mini intel i5 16GB RAM	Virtual box 5.0	Lucidchart
	Vagrant 1.8.1	Moqups
	Heroku	Gantter
	Atom 1.6.0	Taiga
	L <sup>A</sup> T <sub>E</sub> X	
	TexStudio	

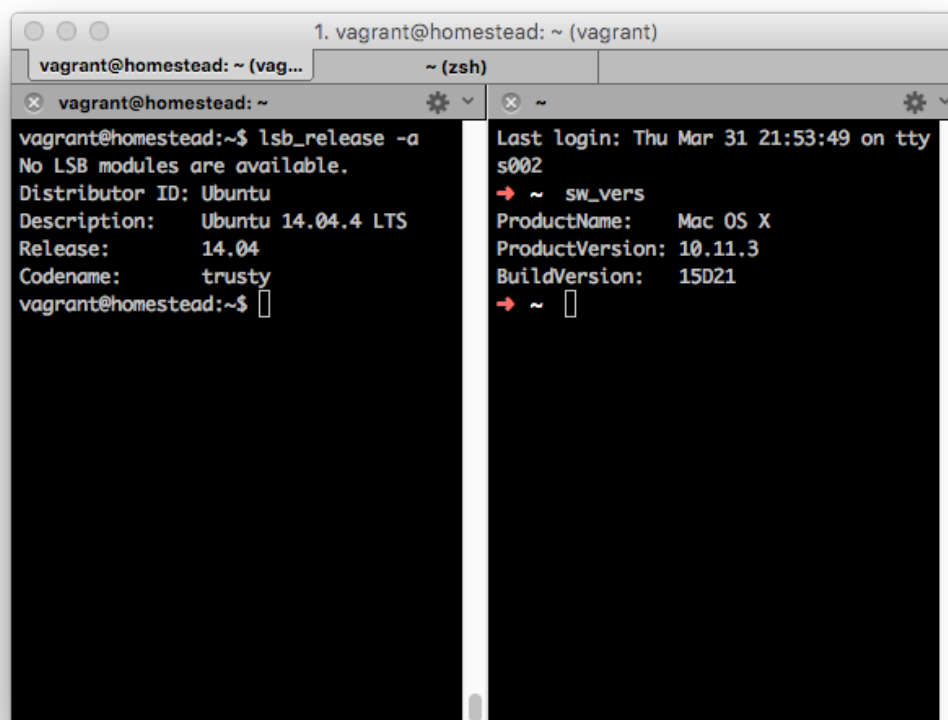
Cuadro 8.1: Herramientas utilizadas.

Lenguajes	Librerías
PHP 7.0	Laravel 7.2
JavaScript 1.8	jQuery 2.2.2
CSS 3	Bootstrap 3.3.6
HTML 5	Font Awesome 4.5.0

Cuadro 8.2: Lenguajes y Librerías.

El desarrollo se ha llevado a cabo en un ordenador Mac mini con una imagen virtualizada del sistema operativo *Ubuntu* sin entorno gráfico y con un conjunto preconfigurado de herramientas para desarrollar en PHP con el *framework Laravel*. No se ha ejecutado esta imagen virtualizada directamente, sino que se ha conectado a ella mediante SSH<sup>1</sup>, lo cual nos ha permitido disfrutar de las ventajas de un entorno de programación correctamente configurado pero al mismo tiempo haciendo uso de nuestras herramientas locales preferidas.

<sup>1</sup>[https://en.wikipedia.org/wiki/Secure\\_Shell](https://en.wikipedia.org/wiki/Secure_Shell)



```
1. vagrant@homestead: ~ (vagrant)
vagrant@homestead: ~ (vag... ~ (zsh)
vagrant@homestead: ~
vagrant@homestead:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:   Ubuntu 14.04.4 LTS
Release:      14.04
Codename:     trusty
vagrant@homestead:~$ 
Last login: Thu Mar 31 21:53:49 on tty
s002
~ sw_vers
ProductName:   Mac OS X
ProductVersion: 10.11.3
BuildVersion:  15D21
~
```

Figura 8.1: El software Vagrant nos permite ejecutar un entorno Ubuntu preconfigurado para desarrollo desde OSX.

La aplicación se ha desplegado en la plataforma *Heroku*<sup>2</sup>, la cual ha servido no solo como hosting sino también como control de versiones mediante *GIT*, siendo este control gestionado por la herramienta de línea de comandos con el mismo nombre. Durante el desarrollo se ha hecho uso de varias librerías de gran popularidad como *jQuery* o *Bootstrap*. Se ampliará información sobre estas en el apéndice número 5.

Son particularmente importantes las herramientas *Ganttter* y *Taiga*. La primera se ha utilizado para la gestión global del proyecto: Planificación, hitos, recursos y riesgos. La segunda se ha utilizado como gestora del método *Scrum*, registrando problemas, mejoras e historias de usuario para la aplicación y gestionando los *Backlogs* y *Sprints*. Dado que la lista de usuarios para los test de usabilidad es muy corta, no se les solicitará que creen una cuenta en *Taiga* para registrar los problemas, sino que estos serán comunicados por cualquier otro canal y serán insertados en la herramienta posteriormente.

Se han utilizado también varias herramientas adicionales. *Lucidchart* para elaborar los gráficos de *workflow* y *Moqups* para los *wireframes*. El presente documento

<sup>2</sup><http://heroku.com>

se ha redactado utilizando L<sup>A</sup>T<sub>E</sub>X con el editor *TeXstudio*.

# Capítulo 9

## Planificación

A continuación se detalla el listado de tareas necesarias para completar el proyecto. Existe un diagrama de Gantt reflejando esta planificación en la siguiente url: <https://goo.gl/0JIFm4>. El diagrama se ha realizado con la aplicación Gantter, disponible en <http://gantter.com>

- Redactar la lista de requisitos para la aplicación
- Decidir las tecnologías ideales para llevarla a cabo
- Investigar conceptos teóricos para apoyar la idea de la aplicación
- Redactar la PAC1 y preparar entregables si los hay (8/3/2016)
- Crear wireframes de la aplicación
- Crear gráficos de *workflow*
- Diseñar la base de datos
- Estudiar documentación de *Laravel*
- Crear página de *login* con funciones de crear usuario con dos perfiles, paciente y doctor
- Crear copia remota de la aplicación en *Heroku* e ir sincronizando con regularidad para ahorrarse el paso de desplegar la app completa una vez terminada.
- Al crear un usuario, este debe crearse en la base de datos con la estructura decidida anteriormente en el diseño
- Crear formulario para introducir registro
- Crear página para ver todos los registros introducidos



- Redactar la PAC2 y preparar entregables si los hay (6/4/2016)
- Crear página de perfil de usuario para dar detalles de la enfermedad, nombre del doctor y email para los avisos
- Crear página para añadir evento de calendario
- Crear vista de calendario
- Crear página para añadir recordatorio o considerar fusionarlos en el calendario
- Redactar la PAC3 y preparar entregables si los hay (8/5/2016)
- Crear página para usuario doctor, para ver los registros de sus pacientes asociados
- Revisar el código y refinar funciones cuando sea necesario
- Crear documento con guía de estilo visual
- Aplicar estilo visual o realizar mejoras de interfaz si es necesario
- Grabar video para la entrega final
- Redactar la entrega final y preparar entregables si los hay (20/6/2016)

#### **9.0.4. Imágenes del proceso**

Nombre	Duración	Inicio	Fin	Predecesoras
Redactar la lista de requisitos para la aplicación	6d	01/03/2016	03/03/2016	
Decidir las tecnologías ideales para llevarla a cabo	6d	01/03/2016	03/03/2016	
Investigar conceptos teóricos para apoyar la idea de la aplicación	6d	01/03/2016	03/03/2016	
Redactar la PAC1 y preparar entregables si los hay	16d	03/03/2016	08/03/2016	
Crear wireframes de la aplicación	1d	10/03/2016	10/03/2016	
Crear gráficos de workflow	1d	11/03/2016	11/03/2016	
Diseñar base de datos	1d	12/03/2016	12/03/2016	
Estudiar documentación de Laravel	1d	12/03/2016	12/03/2016	
Crear página de login con funciones de crear usuario paciente y usu	22d	13/03/2016	20/03/2016	
Crear copia remota de la aplicación en Heroku	1d	13/03/2016	13/03/2016	
Confirmar que la creación de usuario respeta el esquema deseado	4d	19/03/2016	20/03/2016	
Crear página de formulario para introducir un registro	19d	21/03/2016	27/03/2016	
Crear página para ver todos los registros introducidos	19d	28/03/2016	03/04/2016	
Redactar la PAC2 y preparar entregables si los hay	28d	28/03/2016	06/04/2016	4
Crear página de perfil de usuario para dar detalles de la enfermedad	7d	08/04/2016	10/04/2016	
Crear página para añadir evento de calendario	13d	11/04/2016	15/04/2016	
Crear vista de calendario	4d	16/04/2016	17/04/2016	
Crear página para añadir recordatorio o considerar fusionarlos con	19d	18/04/2016	24/04/2016	
Redactar la PAC3 y preparar entregables si los hay	40d	25/04/2016	08/05/2016	14
Crear página para usuario doctor, para ver los registros de sus paci	19d	09/05/2016	15/05/2016	
Revisar el código y refinar o arreglar funciones si es necesario	106d	16/05/2016	20/06/2016	
Crear documento con guía de estilo visual	19d	16/05/2016	22/05/2016	
Aplicar estilo visual o realizar mejoras de interfaz si es necesario	19d	23/05/2016	29/05/2016	
Grabar video para la entrega final	22d	13/06/2016	20/06/2016	
Redactar la entrega final y preparar entregables si los hay	22d	13/06/2016	20/06/2016	19

Figura 9.1: Listado de tareas en el diagrama

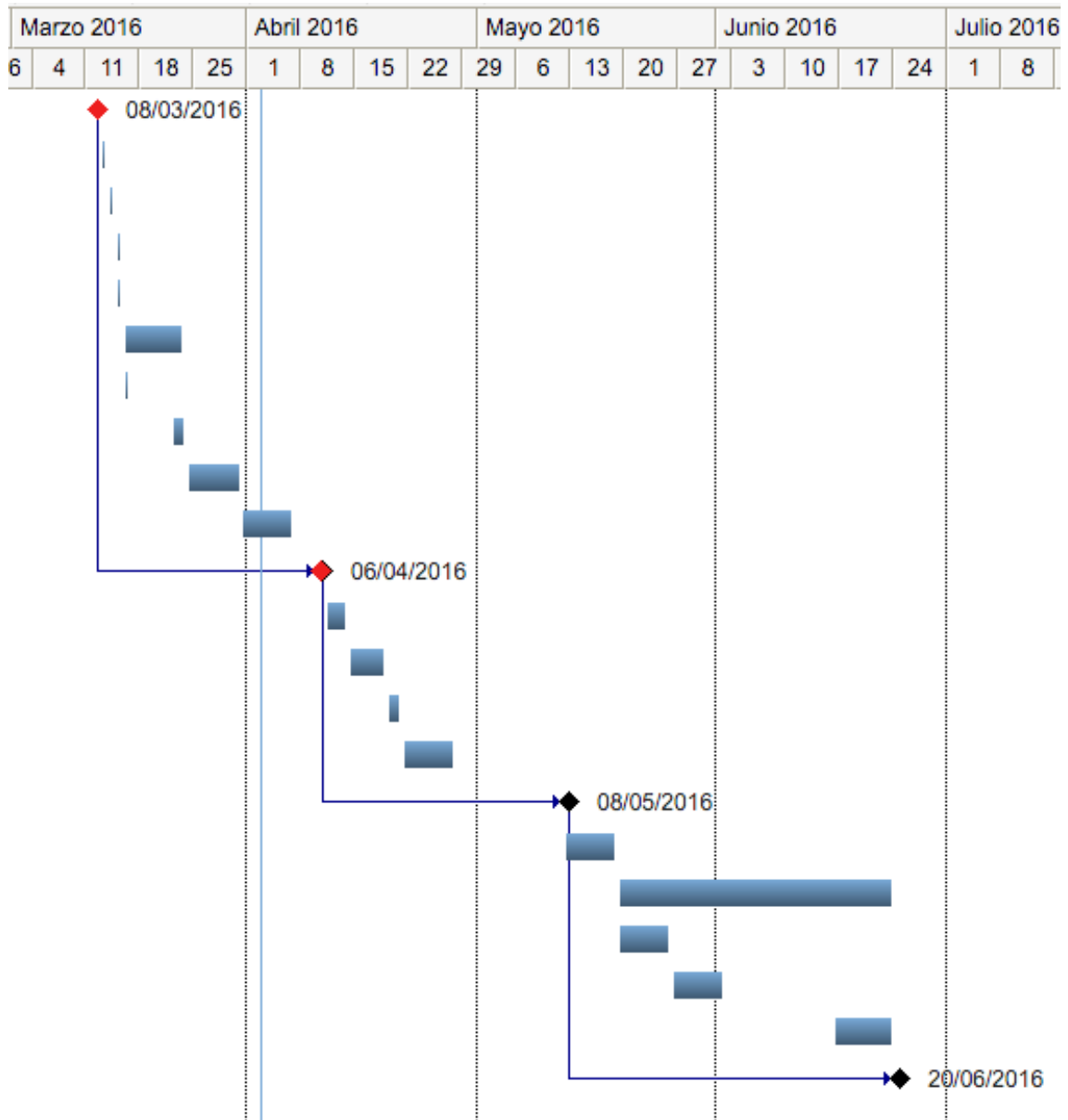


Figura 9.2: Representación gráfica del proceso

# Capítulo 10

## Proceso de Trabajo

Se detalla a continuación el proceso seguido para completar las distintas partes expuestas en la planificación:

**Redactar la lista de requisitos:** Una vez la idea ha sido aprobada por el profesorado de la universidad, se redactan los requisitos estableciendo un equilibrio entre lo que es ideal y lo que puede conseguirse de un modo realista en el plazo de tiempo otorgado.

**Decidir las tecnologías ideales para llevarla a cabo:** Durante el estudio del Grado Multimedia, se ha hecho muy poco hincapié en lenguajes de servidor y nunca a nivel suficientemente avanzado como para realizar gestiones complejas (por ejemplo autenticar al usuario). Actualmente existen multitud de *frameworks* que nos pueden facilitar la tarea, pero incluso la elección del más simple de ellos conlleva un tiempo de estudio y comprensión, lo cual ha de tenerse en cuenta para realizar la elección.

Inicialmente se consideró una combinación del llamado *MEAN Stack*, compuesto por *MongoDB* para la base de datos en formato NoSQL, *Angular.js* como framework MVC para el cliente, *Node.js* para la parte del *backend* y *Express* como *web framework* de *Node*. Esta combinación resultaba la más atractiva, ya que a priori ofrecía un código simplificado (todos los lenguajes eran *Javascript*) y mucha facilidad para construir los servicios REST que deberían alimentar a una hipotética aplicación móvil.

Sin embargo, durante el grado en la universidad no hemos trabajado con ninguna de esas tecnologías, así que el aprendizaje de todas las partes podría llevar demasiado tiempo.

Una vez descartada esa opción, la solución más interesante ha sido *Laravel*, un *framework* de PHP creado por Taylor Otwell en 2011<sup>1</sup> con la finalidad de

---

<sup>1</sup><https://es.wikipedia.org/wiki/Laravel>

disponer de una sintaxis elegante, con separación Modelo Vista Controlador y facilidades adicionales como un motor de plantillas incorporado, gestión de la base de datos simplificada y programación de rutas de manera muy intuitiva. La ventaja es que existían partes que ya se habían visto en asignaturas anteriores (PostgreSQL, jQuery...), aunque la estructura y funcionamiento de Laravel era totalmente nueva, lo cual también requería tiempo de estudio.

Basar todo el proyecto en *Laravel* se presenta como una opción mucho más interesante que utilizar PHP puro añadiendo clases y paquetes de *Composer* para añadidos extra. Basando el proyecto completo, se adquiere un conocimiento global extrapolable a otros proyectos futuros, al mismo tiempo que se ahonda en el proceso de aprender completamente un nuevo método partiendo de cero.

La adición de *Vagrant*<sup>2</sup> al proceso ha sido muy positiva ya que ha permitido disponer de un entorno perfectamente configurado con *PHP*, *Composer*, *PostgreSQL* y toda una selección de herramientas como *Node.js* o *REDIS* que incluso si no se han utilizado en este proyecto representan un gran descubrimiento con perspectiva de utilizarlas en el futuro.

**Investigar conceptos teóricos para apoyar la idea de la aplicación:** La mayor parte de este concepto se trabajó como prólogo al comienzo de la primera PAC. Se han buscado ideas relevantes en varios libros y artículos científicos para apoyar las ideas básicas de la aplicación, como por ejemplo el beneficio de registrar el estado de salud con frecuencia o la idoneidad de utilizar el sistema de caras para medir el dolor. Una vez estas ideas están asimiladas, el marco teórico pasa a un segundo plano y el grueso de la aplicación se centra en crear un producto funcional que cumpla dichos requisitos

**Redactar la PAC1 y preparar entregables si los hay (8/3/2016):** La primera entrega es de gran importancia, ya que sienta las bases de lo que serán las entregas posteriores. Por una parte no existe parte técnica para entregar, pero se debe presentar la planificación, lo cual requiere trabajo extra. Asimismo, hay detalles que han de tenerse en cuenta como la preparación de las plantillas de  $\text{\LaTeX}$ , que pueden llevar más tiempo del esperado si no se planean correctamente. Afortunadamente, una correcta preparación y planificación de esta PAC significará mayores facilidades para las entregas futuras

**Crear wireframes de la aplicación:** El problema principal ha sido encontrar una herramienta para realizar wireframes de manera eficiente y que al mismo tiempo no estuviera severamente limitada por una licencia. La suite de Adobe

---

<sup>2</sup><https://www.vagrantup.com/>

provista por la Universidad no ofrece ninguna herramienta específica para wireframes, si bien es cierto que *Illustrator* o *Indesign* pueden servir para el propósito, no tienen la facilidad y flexibilidad de una herramienta específica.

Se han valorado herramientas profesionales. Por ejemplo, el estándar de la industria *OmniGraffle*, descartado por su precio, motivo que también ha dejado a un lado alternativas como *Moqups* o *Balsamiq*. Al final la elección ha recaído en *NinjaMock*<sup>3</sup>, un programa con características muy similares a *Moqups* pero incluyendo la exportación en .png, que en el resto de alternativas suele estar incluida sólo en la opción de pago.

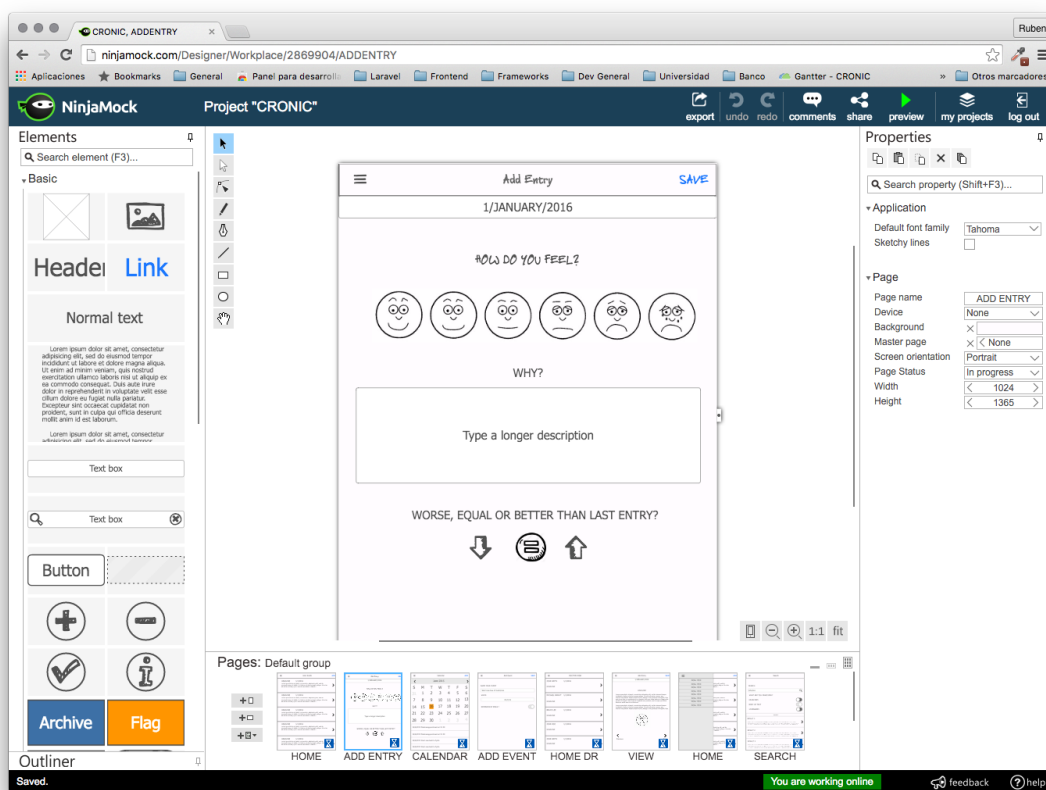


Figura 10.1: Interfaz de NinjaMock

**Crear gráficos de *workflow*:** Este punto se ha visto beneficiado de lo aprendido en la asignatura Arquitectura de la Información, lo cual ha resultado clarificador y de utilidad para desembocar en el diagrama final. Adicionalmente hay que reseñar que la estructura planteada no presenta gran complejidad, así que esta tarea no resulta nada problemática

**Diseñar la base de datos:** Punto inicialmente previsto cuando la idea inicial sugería diseñar una BD relacional y plasmarla en *PostgreSQL*. El cambio a *Lara-*

<sup>3</sup><http://ninjamock.com/>

*vel* ha traído consigo el esquema *Eloquent*, posibilitando gestionar las bases de datos de una manera muy distinta a lo pretendido originalmente. Por ello, gran parte del tiempo destinado a diseñar la BD relacional se dedica a estudiar cómo funciona *Eloquent* y de qué forma puede simplificar nuestras interacciones.

**Estudiar documentación de *Laravel*:** Uno de los puntos cruciales de todo el trabajo. Partiendo de un conocimiento inexistente sobre este *framework* sumado a un conocimiento muy pobre de *PHP*, este trabajo se plantea como una herramienta para aprender *Laravel* desde cero utilizando un proyecto de ejemplo muy atrayente.

**Crear página de *login*:** Una vez generado el esqueleto de la aplicación, el primer paso es crear una estructura en la cual todos los pasos se lleven a cabo dentro del contexto de usuario registrado. Para ello *Laravel* incorpora un modelo de autenticación que viene prácticamente configurado de serie. Lógicamente, su configuración de prueba no nos sirve, ya que hay que adaptar el usuario a los requisitos de nuestra aplicación. Por ejemplo, en el registro ha de especificarse si la persona registrándose es un paciente o un doctor, con lo cual hay que modificar tanto las plantillas de registro como los esquemas de la base de datos.

Por suerte, el proceso no es difícil, aunque en un principio puede parecer confuso por las diferencias entre el adaptar una base de datos normal (por ejemplo, hacer un `ALTER table` en *SQL*) y modificarlo aquí, lo que conlleva rectificar las migraciones y utilizar la aplicación *artisan* para ejecutarlas.

Una vez se resuelve este asunto, nos encontramos ante un esqueleto provisto de autenticación, sobre el cual ya podemos empezar a construir nuestra aplicación.

**Crear copia remota de la aplicación en *Heroku*:** Si la versión definitiva de la aplicación va a estar desplegada en *Heroku*, es razonable comenzar a desplegarla cuanto antes. De ese modo se evitan hipotéticos problemas que pudieran ocurrir al final del proceso, al intentar subir una aplicación de gran tamaño en la cual es más difícil identificar de dónde vienen los errores. Comenzando a desplegarla desde su inicio, en cada *commit* que hagamos en *GIT* podemos acotar el terreno de manera mucho más refinada para identificar los posibles fallos. A esto también ayuda la facilidad de desplegar en *Heroku*, lo cual puede hacerse desde su propio servicio de *GIT*, desde un repositorio de *GitHub* o incluso desde una carpeta de *Dropbox*. En el caso que nos ocupa se ha utilizado el propio servicio *GIT* de *Heroku*.

**Revisar que la estructura de la BD creada se ajusta al diseño:** Este paso se

planeó inicialmente antes de decidir el paso a *Laravel*, con lo cual una implementación de tablas en la base de datos era mucho más compleja y requería de una revisión más cuidadosa para confirmar que el esquema se ajustaba al comportamiento que esperábamos.

Al utilizar *Laravel* y *Eloquent*, la gestión se ve facilitada sobremanera, con lo cual la revisión deja de cobrar tanto protagonismo, ya que el sistema se encarga de muchas de las tareas que anteriormente tendríamos que haber revisado manualmente.

**Crear formulario para introducir un registro:** Se crea el formulario para registrar en la base de datos una entrada del paciente. En una primera etapa se añaden los datos no estructurales como título o texto. En el futuro se implementará un sistema para poder insertar datos estructurales, ofreciendo al usuario flexibilidad para elegir los mismos, ya que no todos tendrán las mismas necesidades (presión arterial, peso, nivel de azúcar, etc...)

**Crear página para ver todos los registros introducidos:** El listado de todas las entradas es un elemento en el cual de haberlo realizado en *PHP* puro hubieran surgido varias dificultades, como por ejemplo la paginación (que siempre es algo más compleja sin librerías auxiliares). *Laravel* facilita esto y otras funciones necesarias. En una primera etapa se muestran todas las entradas sin filtros. En una versión futura, se implementarán filtros de búsqueda.

**Redactar la PAC2 y preparar entregables si los hay (6/4/2016):** La clave de la segunda PAC es el ser la primera entrega en la cual hay que presentar una versión del producto. Se adjuntará una versión preliminar alojada en *Heroku* junto con la dirección del repositorio del código. La versión presentada incluye:

- Posibilidad de registro de usuario
- Introducción de entradas en el diario incluyendo la escala de dolor
- Vista general de todas las entradas
- Vista "Panel de control" con la última entrada destacada

El resto de elementos se incorporarán en breve después de la segunda entrega. También se adjunta un documento adicional con todas las imágenes de los *Wireframes* y bocetos preliminares a tamaño completo, los cuales también se presentan resumidos en el capítulo pertinente de esta memoria

**Cambio en la metodología:** Durante la realización de la PAC2 usando la metodología *Scrum*, se valoran otras alternativas. *Scrum* es muy poderoso para



trabajos en equipo, pero para una idea simple como la que nos ocupa, el tiempo en mantener las pizarras de los sprint no compensa con respecto al beneficio que produce en un proyecto de una sola persona.

Debido a que no es posible aprovechar al máximo las ventajas de *Scrum* para el trabajo en equipo, ya que no hay equipo, se opta por pasar al método *Kanban*, que se compone de una única pizarra con varios apartados en los cuales hay que mover pequeñas tarjetas con las tareas pendientes según si están en proceso, listas para probar o finalizadas por completo.

El planteamiento es mucho más simple, y tras trabajar con el sistema durante varias semanas, decide adoptarse para el resto de tareas pendientes hasta la finalización del trabajo.

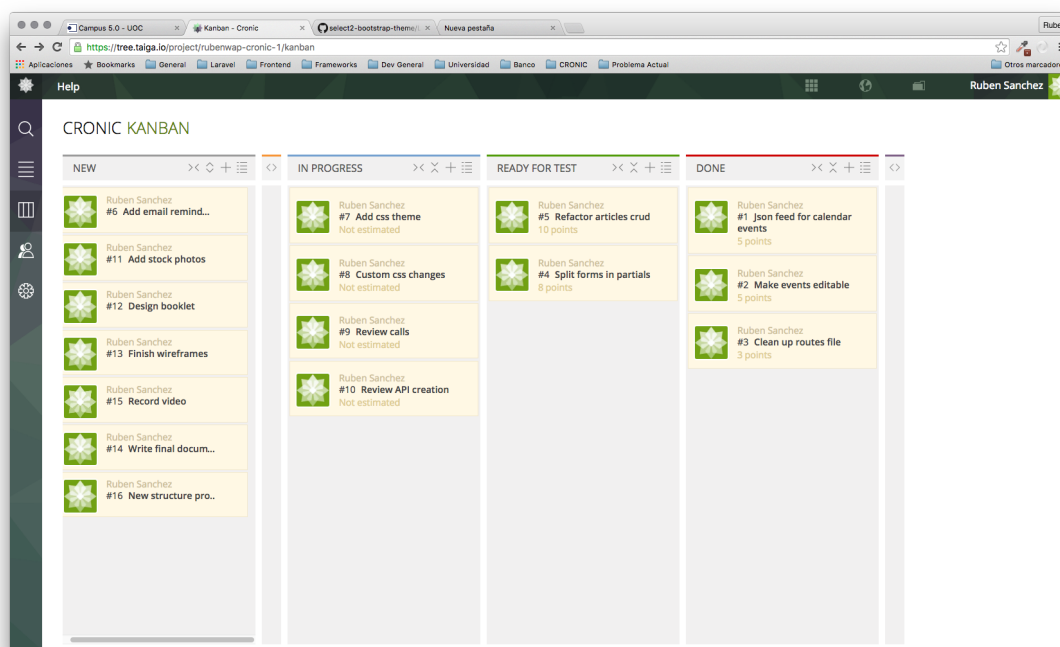


Figura 10.2: Vista de la pizarra única de Kanban.

**Crear página de perfil de usuario:** Se crea una página donde además de datos triviales como peso y estatura, pueden seleccionarse alergias, detallar la enfermedad o enfermedades del usuario y especificar qué doctores la están tratando. Para esto, se incorpora una rutina a las funciones de la página que explora el listado de usuarios registrados y extrae aquellos que se han registrado con un perfil de doctor. En el futuro, los doctores podrán acceder a un panel de control en el cual podrán leer las entradas compartidas de sus pacientes.

Para conseguir esta funcionalidad, se han utilizado varias librerías javascript para la parte visible al usuario, ya que las alergias y los doctores se selec-

cionan de una lista predeterminada, y los elementos seleccionados pueden ser múltiples. A nivel de *backend*, se resuelve utilizando sintaxis de *Eloquent*.

Lamentablemente un fallo de diseño a la hora de trazar las relaciones entre paciente y doctor en la base de datos, ha dificultado sobremanera la realización de la vista del panel de control en la cual el usuario doctor puede ver las entradas de diario de sus pacientes. Este fallo será subsanado en un futuro, aunque fuera del ámbito de la entrega de este proyecto.

**Crear página para añadir evento de calendario** Para esta función se pretendía utilizar la librería *Laravel-FullCalendar*, que facilita la creación de eventos compatibles con *FullCalendar* (ver siguiente punto), pero no ha sido necesario. *FullCalendar* simple sin añadidos es lo suficientemente accesible como para utilizarlo sin complementos. La instalación de la librería del plugin de jQuery permite recibir los datos sobre los eventos mediante un *feed* de *JSON*, con lo cual sumado a las capacidades de *Laravel* para crearlo, obtenemos como resultado un calendario con eventos realizado de manera simple y efectiva.

**Crear vista de calendario** Una vez añadido el plugin de jQuery *FullCalendar*<sup>4</sup>, un calendario al cual pueden añadirse eventos dinámicamente, encontramos un problema: Después de haber añadido las rutas y funciones para editar y borrar artículos, el archivo de rutas comienza a ser bastante confuso, a causa de no seguir los patrones establecidos de *Laravel* para rutas. Llegado a este punto, se hace necesaria una reformulación de todo el sistema de edición, para así poder permitir no solo editar y borrar artículos, sino también editar y borrar eventos de calendario. Esto se consigue gracias a la unificación de rutas en recursos y a una simplificación en los formularios gracias al uso de páginas parciales

**Crear página para añadir recordatorio:** Lo que antes iban a ser dos funciones distintas (calendario y recordatorios) se fusionarán en una para evitar confusiones. Al fin y al cabo, el intentar diferenciar qué merece una entrada en el calendario y qué merece un recordatorio, puede resultar confuso para el usuario, así que este punto se cancela.

**Redactar la PAC3 y preparar entregables si los hay (8/5/2016):** La importancia de esta PAC ha radicado en el hecho que la funcionalidad presentada debía ser prácticamente final. Finalmente se consigue llegar a un grado de funcionalidad interesante en el cual prácticamente todo lo planeado se encuentra presente, salvo las opciones de visualización para los doctores. El redactado

---

<sup>4</sup><http://www.fullcalendar.io>

de la memoria necesita incluir algunos apartados para reflejar los cambios en metodología desde la PAC anterior.

**Crear página para usuario doctor:** Este punto es quizá de los más problemáticos que existen en la aplicación, ya que es necesario aplicar técnicas de *Eloquent* avanzadas para gestionar la aparición de información de un usuario en el perfil de otro usuario. El problema básico radica en que la base de datos se diseñó con un problema que al inicio pasó desapercibido. Actualmente se opta por no solucionar la vista doctor, ya que otras partes son más prioritarias.

**Añadir formularios AJAX:** Un punto importante que mejora la usabilidad de la aplicación es la posibilidad de introducir entradas tanto de diario como de calendario vía peticiones AJAX. Hasta el momento todas las peticiones eran convencionales, lo cual conlleva que ante cada edición o añadido fuera necesario actualizar la página. Se incluyen las siguientes mejoras:

- Formularios de entrada rápida de contenido desde el panel de control principal
- Borrado rápido de entradas desde la vista listado
- Edición rápida desde la vista listado
- Edición del perfil de usuario y actualización de la información en vivo.

Se decide mantener los formularios convencionales en otros apartados, y al mismo tiempo se crea una función para garantizar que incluso si la petición AJAX falla por razones de lentitud o inaccesibilidad del servidor, el contenido del usuario se intentará enviar de nuevo automáticamente a través de una petición convencional.

**Crear documento con guía de estilo visual:** Esta tarea comenzó siendo un documento separado con los diseños, aunque en la entrega final se integran en un apéndice de esta memoria.

**Aplicar estilo visual o realizar mejoras de interfaz si es necesario:** Se está utilizando *Bootstrap* como base para el estilo *CSS*. Una vez la parte de programación está finalizada, se crean adaptaciones en un archivo *css* separado para crear un estilo visual propio.

**Grabar video para la entrega final:** Utilizando *Audacity* para el audio y *AfterEffects* para el video, se prepara una defensa visual del proyecto.

**Redactar la entrega final y preparar entregables (20/6/2016):** Por último se entrega la compilación de archivos que comprende la entrega final, esta vez incluyendo todo el material disponible, el producto finalizado.

# Capítulo 11

## Prototipos

Se adjuntan las imágenes de los prototipos utilizados para el diseño inicial. Algunas distribuciones han cambiado ligeramente en comparación con el resultado del producto final (ver imágenes en el apéndice Capturas de Pantalla). En un producto donde hay varias personas involucradas, el arquitecto de la información debe actualizar los *wireframes* cada vez que hay cambios.

En el contexto de un proyecto de una sola persona, tener una versión inicial es una buena herramienta para entender cómo estructurar la aplicación, pero no es necesario preparar una nueva versión con cada cambio, ya que no aporta ninguna mejora al proceso de trabajo, al ser el programador y el arquitecto la misma persona.

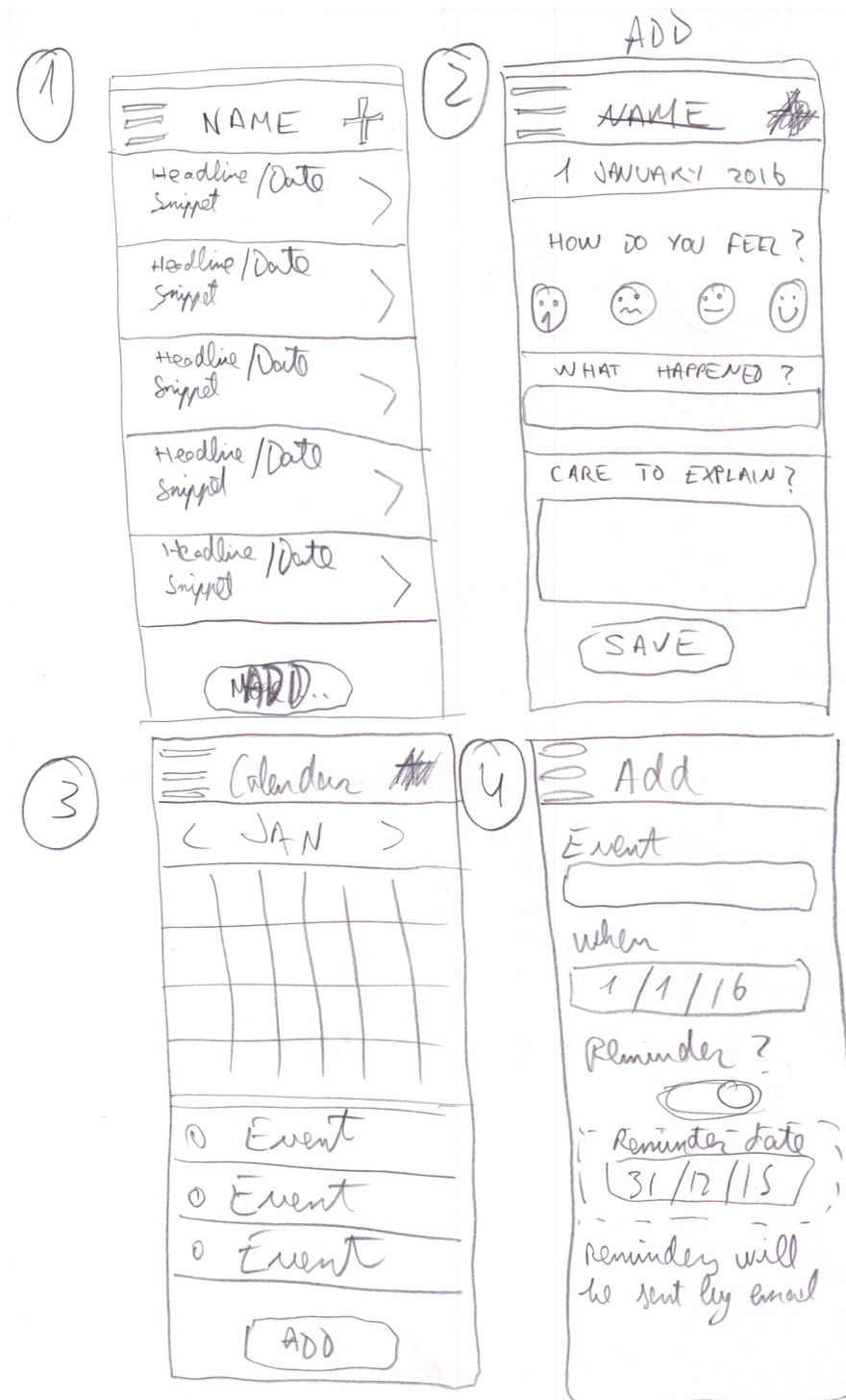



Figura 11.1: Algunos bocetos preliminares anteriores a la realización de los prototipos.

☰	Add Entry	SAVE
1/JANUARY/2016		

HOW DO YOU FEEL?



WHY?

Type a longer description

WORSE, EQUAL OR BETTER THAN LAST ENTRY?




Figura 11.2: Protitipo de la pantalla de introducción del diario

☰ Add Event SAVE

**NAME YOUR EVENT**

**WHEN**

**REMINDER BY EMAIL?**

Figura 11.3: Prototipo de la pantalla de introducción de recordatorios. La opción de email no ha sido incorporada en la versión final





☰	Add Entry	SAVE
1/JANUARY/2016		
<h3>HEADLINE</h3> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis</p> 		
<		>
Previous		Next

Figura 11.4: Prototipo de la pantalla de vista del diario

 Settings SAVE

**YOUR NAME**

**YOUR EMAIL (used for reminders)**

**NAME YOUR ILLNESS**

**FIND YOUR DOCTOR**




Greg_ 
Gregory House
Gregorio Holtzman
Gregor Samsa

Figura 11.5: Prototipo de la pantalla de editar perfil

Search

SEARCH



WHAT ARE YOU SEARCHING?

HEADLINES


BODY OF TEXT

USERNAMES

RESULTS


**RESULT 1**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit



**RESULT 2**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit



**RESULT 3**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit




Figura 11.6: Prototipo de la pantalla de búsqueda, no incluida en esta versión.

# Capítulo 12

## Perfiles de Usuario

En el siguiente apartado se detallarán los "personajes de usuario" (*user persona* en inglés). Puede encontrarse una descripción sobre la interacción y los test con usuarios en el capítulo siguiente: *Usabilidad*.

### 12.1. Personajes de usuario

Se introducen a continuación dos perfiles distintos que representan un gran porcentaje del usuario modelo al cual se enfoca la aplicación.

#### 12.1.1. Personaje 1: Roberto, 26 años

Roberto trabaja como periodista de nuevas tecnologías para un medio de comunicación. Por ello, está habituado a utilizar todo tipo de aparatos e innovaciones tecnológicas. Tras el diagnóstico de una enfermedad de piel crónica, Roberto encuentra que tiene dificultades para asociar los días de mayor o menor irritación y dolor con las medicinas tomadas, ya que el doctor varía la dosis y tipología de estas con cierta frecuencia.

Utilizando Cronic, Roberto toma anotaciones relacionando el nivel de dolor con las medicinas tomadas. Igualmente, anota una breve descripción de otros síntomas generales. Utiliza la aplicación desde varios dispositivos: teléfono móvil, tablet u ordenador según el momento del día.

#### 12.1.2. Personaje 2: Sonia, 58 años

Sonia trabaja como camarera de pisos en un hotel. El psicólogo le ha diagnosticado un trastorno de personalidad. Ya que no está acostumbrada a las nuevas tecnologías, ha sido su hija adolescente quien le ha mostrado el funcionamiento de la aplicación.

Sonia utiliza teléfono móvil, pero únicamente envía mensajes, con lo cual siempre utiliza la aplicación desde el ordenador de casa.

Escribiendo una vez al día, Sonia no solo añade los datos indicados por el psicólogo, sino que aprovecha el apartado de texto libre para utilizarlo como un diario genérico.

# Capítulo 13

## Usabilidad

La usabilidad de la aplicación debería depender mucho de la disposición de elementos en los prototipos, que al mismo tiempo está muy influida por la disposición por defecto en el *framework CSS* utilizado, que a su vez recibe influencias de las convenciones y hábitos más frecuentes en el ámbito de diseño de interfaces.

El método ideal para comprobar la idoneidad de esta distribución es el test de usuarios.

En el capítulo anterior se han descrito dos perfiles de usuario que corresponden a los dos usuarios que se someterán al test. No son las mismas personas exactas, pero su perfil es muy similar.

### 13.1. Partes del test

El test consta de dos partes. En la primera, tras una breve explicación sobre para qué sirve la herramienta, se deja que el usuario la utilice de manera libre mientras es observado. En la segunda parte se les solicita su opinión.

#### 13.1.1. Perfil Roberto

Roberto llega a la prueba con un *iPad Mini*, ya que su test se realizará sobre esta plataforma. Las conclusiones extraídas son las siguientes:

- Existen algunos fallos graves con la estructura responsiva del CSS. Durante el desarrollo no se ha testado suficiente, pero algunas pantallas muestran la estructura para escritorio pese a visualizarse desde una tablet de tamaño pequeño.
- Hay otros fallos de usabilidad menores, como algunos botones demasiado pequeños para ser pulsados con un dedo en vez de un ratón. Un ejemplo son las *checkbox*.

- El usuario echa de menos más apartados para añadir información estructurada en las entradas, como presión sanguínea, etc.
- Hay cierta confusión por tener dos pantallas distintas para la introducción de entradas y eventos. Se sugiere unificarlo todo y ofrecer una única vía de introducción
- Se sugiere añadir un filtro para modificar las fechas mostradas en el gráfico del dolor
- Se sugiere añadir una función de búsqueda

### 13.1.2. Perfil Sonia

Sonia realizará la prueba en un ordenador de escritorio. Las conclusiones extraídas son las siguientes:

- Cuando la página se muestra en disposición de escritorio (los títulos de los apartados desaparecen en el menú lateral), resulta confuso recordar dónde está cada apartado. Se sugiere añadir los nombres de las secciones también en esta disposición.
- Tras descubrir el botón de añadir una entrada rápida desde el panel de control, se intenta pulsar al botón con el mismo icono en la barra lateral, y sorprende que lleve a una página distinta (introducción de la entrada a pantalla completa en vez de con un popup modal)
- El popup modal crea confusión porque al contrario que en la mayoría de páginas, hacer clic fuera de su superficie no lo desactiva, teniendo que pulsar el botón de cerrar en la esquina superior derecha.
- No parece obvio que el carrusel de imágenes de caras de dolor se selecciona simplemente moviéndose a la cara apropiada. La usuario intenta hacer clic en la cara para confirmar que esa es la elegida. Se echa de menos más claridad en este aspecto

### 13.1.3. Conclusiones

Se realiza una lista de tareas pendientes a mejorar. Debido al calendario de entrega de la aplicación, no va a ser posible finalizarlas todas antes de esa fecha. La lista es la siguiente:

- Retocar las resoluciones de los media query de CSS, para que las distribuciones de pantalla se muestren correctamente en los dispositivos apropiados

- Sustituir los *checkbox* por interruptores tipo
- toggle
- Reconsiderar la necesidad de ofrecer dos páginas distintas para la introducción de entradas
- Añadir una función para cerrar el modal al pulsar fuera de este
- Aclarar en el formulario de entradas que la cara visible es la cara seleccionada
- Considerar los textos del menú en la disposición de escritorio
- Añadir función de búsqueda
- Añadir filtro para las fechas del gráfico de dolor



# Capítulo 14

## Seguridad

Al ser una aplicación basada en *Laravel*, la protección de *Cronic* está estrechamente ligada a las características de seguridad que ofrece dicho *framework*. Dos de las más prominentes son:

- Previene inyección SQL mediante PDO (Eloquent realiza la petición por defecto de este modo)
- Previene el *Cross Site Scripting* mediante el escape de caracteres.

Y por supuesto, se le añade la capa de seguridad de Heroku:

- Firewalls
- Mitigador de DDoS
- Protección contra textitspoofing
- Protección contra escaneo de puertos
- Cada aplicación se ejecuta en un entorno aislado y no puede interactuar con el resto de aplicaciones.
- El acceso a las BBDD PostgreSQL requiere encriptación SSL.

Cuando la complejidad del proyecto vaya en aumento, puede ser necesario revisar las medidas de seguridad.

# Capítulo 15

## Versiones

Al ser una aplicación web, las versiones se plantean bajo el sistema de liberación continua (*rolling release*). A excepción de las tres versiones entregadas durante las PAC, que según los requisitos debería corresponder a alpha, beta y final, el resto de versiones corresponderán únicamente a las actualizaciones de código que se muevan al servidor remoto semanalmente con motivo de los *sprint*.

El único indicador fiable para etiquetar las versiones en el servidor son los números de referencia de los *commit* de *Git*. La versión final presentada es el commit número 51.

# Capítulo 16

## Instrucciones

Como cualquier aplicación web moderna, no se redactan instrucciones de usuario, ya que se presupone que el trabajo de usabilidad y diseño de interfaz es suficientemente claro como para que se entienda la funcionalidad sin necesidad de leer un documento.

La única opción razonable es la del uso de *overlays* de ayuda sobre la pantalla la primera vez que el usuario entra tras registrarse. Existen algunas librerías interesantes como *Impromptu*<sup>1</sup> o *Chardin*<sup>2</sup>

Se plantea utilizar estas librerías en el futuro cuando la aplicación se ofrezca al público, pero se descarta utilizarlas para la entrega final.

---

<sup>1</sup><http://trenrichardson.com/Impromptu>

<sup>2</sup><http://heelhook.github.io/>

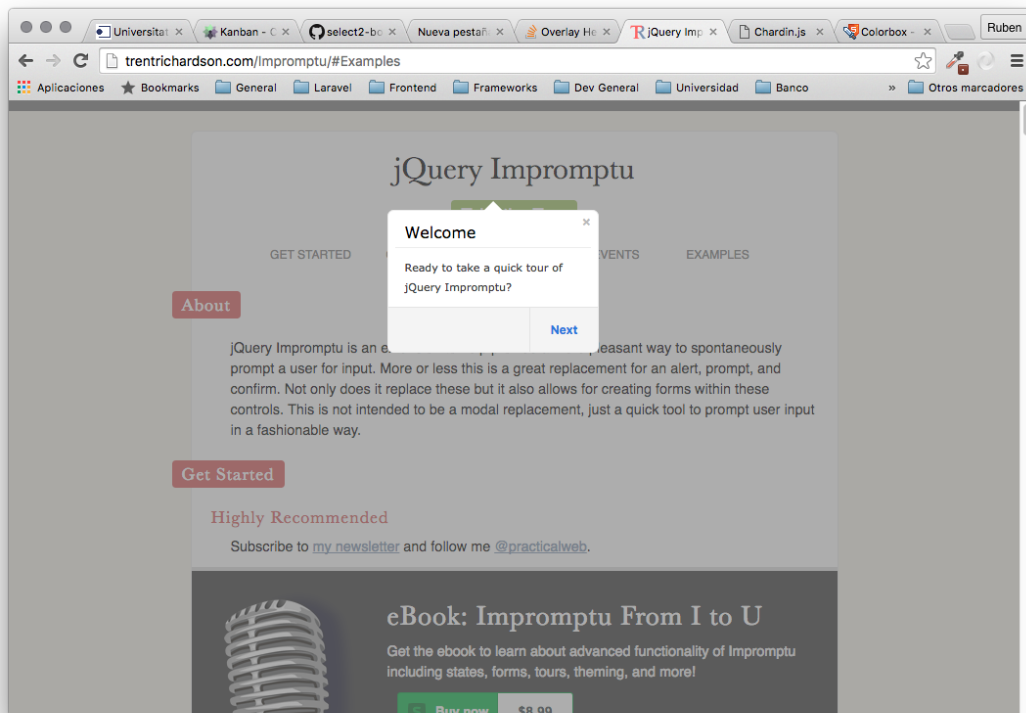


Figura 16.1: Ejemplo de Impromptu



Figura 16.2: Ejemplo de Chardin

# Capítulo 17

## Bugs conocidos

En la versión final presentada, existen los siguientes bugs.

- La pantalla de ver artículo no muestra correctamente los saltos de línea. Estos están guardados en el texto (así se puede ver si se edita un artículo), pero a la hora de mostrarse existen fallos.

Y además falta por implementar las siguientes funciones:

- Añadir la opción de enviar recordatorio por email para los eventos
- Permitir añadir campos personalizados a la pantalla de crear entrada
- Implementar la vista para doctores
- Implementar pantalla de búsqueda
- Implementar filtros de fechas para los gráficos.

# Capítulo 18

## Proyección de Futuro

Una vez entregada la versión final del trabajo, se contempla el continuar desarrollando funcionalidades hasta convertirlo en un producto viable. Para ello, se esquematiza una serie de pasos:

- Completar todos los requisitos de código iniciales. Principalmente se le da importancia a la opción de vista para doctores, ya que es la característica más notable que no ha sido posible incluir en la entrega final.
- Ampliar la capacidad de hospedaje, ya que actualmente la aplicación se encuentra alojada en pruebas, lo cual es inviable para un producto público
- Profundizar en el estudio de mercado, entender con más exactitud qué productos similares existen y en qué lugar podría encajar esta aplicación
- Una vez decidido, trazar un plan de acción que encaje con esa idea (por ejemplo, si es prioritaria una aplicación móvil nativa, si es más importante añadir una funcionalidad u otra, etc.)
- Como alternativa, incluso si se opta por la no continuación del proyecto en su faceta más convencional, se publicará el código fuente en *GitHub* y se planteará un crecimiento como proyecto comunitario en vez de oportunidad comercial.

# Capítulo 19

## Presupuesto

Atendiendo a las características detalladas de la aplicación, tal y como se han especificado a lo largo del siguiente texto, se plantea un posible presupuesto del coste real de esta si tuviera que ser programada desde cero por terceros:

Concepto	Precio en Euros
Diseño Gráfico de la web	150
Programación de la aplicación y sus funcionalidades	500
Maquetación del estilo a la imagen del diseño gráfico	300
Registro del dominio y alojamiento por un año	120
Servicios de Marketing por Internet (opcional)	250

Cuadro 19.1: Presupuesto.

- Total con márketing: Mil trescientos veinte euros.
- Total sin márketing: Mil setenta euros..

A estos precios es necesario sumarles el IVA.

# Capítulo 20

## Análisis de mercado

### 20.1. Herramientas actuales

Un motivo por el cual las aplicaciones dedicadas al registro de la enfermedad no han despegado totalmente, puede ser debido a la popularidad conseguida por las herramientas genéricas para publicar blogs. En algunos casos, plataformas de blogs genéricos dedican páginas específicas a realizar publicidad sobre las posibilidades de registrar la salud del usuario<sup>1</sup>. Podemos también encontrar otras plataformas aparentemente más especializadas en ese campo<sup>2</sup>, pero la gran mayoría de artículos informales que recomiendan llevar este tipo de registros, suelen sugerir herramientas genéricas o incluso papel<sup>3</sup>.

#### 20.1.1. Aplicaciones móviles

Con la llegada de las aplicaciones móviles, apareció un escenario en el cual surgían aplicaciones de una variedad difícilmente vista en otras plataformas anteriores. Particularmente esta gran variedad se notó en la plataforma *Android*, debido a la facilidad de publicar en ella.

Sin embargo, tras examinar el listado de aplicaciones de este género en la tienda *Google Play* y en la *App Store* con un filtro que excluya los productos de pago, podemos observar que la gran mayoría de diarios de salud se dirigen a conceptos muy concretos, como por ejemplo el ejercicio físico y la dieta o los registros del dolor en casos de dolor crónico.

El número de aplicaciones que pretenden registrar y gestionar entradas médicas sin importar la enfermedad en cuestión es extremadamente limitado, lo cual nos abre la puerta para presentar un desarrollo de calidad que pueda alcanzar al mayor

---

<sup>1</sup><https://penzu.com/health-diary>

<sup>2</sup><http://www.mytherapyjournal.com>

<sup>3</sup><http://lifehacker.com/why-you-should-keep-a-journal-and-how-to-start-yours-1547057185>



número de usuarios relevantes posibles.

# Capítulo 21

## Conclusiones

Las siguientes conclusiones preliminares pueden variar en la entrega final de este documento.

### 21.1. Conclusiones sobre temas teóricos

El realizar una aplicación dirigida a un tipo de público tan específico ha servido para enfatizar lo importante que es planear el diseño en concordancia con el usuario. No sabemos qué posibles enfermedades podemos encontrar en quienes utilizan nuestro producto, así que es necesario tener en cuenta todos los factores de accesibilidad posibles. Daltonismo, problemas de movilidad, etc. Este punto es algo que no siempre se tiene en cuenta en todos los desarrollos, pero trabajar en este proyecto ha servido para comprender su importancia.

Igualmente, debido a ser la primera vez que se afrontaba una tarea de este tamaño, ha resultado interesante reflexionar sobre los aspectos teóricos en la planificación de proyectos que se han enseñado en la UOC a lo largo de todo el grado. Sin una planificación lógica y razonada, ordenar y dar prioridad a la multitud de pequeñas tareas que comprenden el cuerpo final, hubiera sido mucho más difícil.

### 21.2. Conclusiones sobre temas prácticos

Las conclusiones más interesantes son sobre asuntos prácticos. Para una persona que nunca antes había elaborado una aplicación completa, el pasar de los pequeños ejercicios vistos en el grado a algo de este calibre, supone un gran paso hacia adelante, afianzando la capacidad de afrontar retos similares en el futuro.

Durante la elaboración de *CRONIC*, se han visto tecnologías y métodos de gran utilidad, no solo a nivel teórico o práctica con pequeños ejemplos, sino en el contexto de un proyecto de gran tamaño, lo cual aporta una perspectiva y un conocimiento

de mucha utilidad con vistas al futuro.

Algunas de las tecnologías aprendidas especialmente para este trabajo, y lo que han aportado:

**Heroku:** El configurar los servidores, bases de datos y demás programas necesarios para ejecutar una aplicación web, ya no es un requisito indispensable. Con la llegada de productos de *Platform as a service* como Heroku, el desarrollador puede centrarse en programar su aplicación, sin temor a fallos en la configuración del servidor o problemas con los permisos de la base de datos. Un servicio extremadamente simple que ha sido a su vez un gran descubrimiento. Su uso es tan corriente a día de hoy que incluso debería mencionarse en algunas asignaturas relevantes del Grado Multimedia.

**Laravel:** Con Laravel se confirma que una librería es la forma más eficiente de exprimir al máximo un lenguaje como PHP sin necesidad de tener un conocimiento experto de este. Por supuesto ha sido necesaria una fase de aprendizaje, especialmente porque el modelo MVC era algo que no se había enseñado durante el grado, lo cual tiene a causar cierta confusión cuando se ve por primera vez. Tras un proceso de lectura de documentación, ha sido posible construir la estructura del programa en *Laravel*, siendo el resultado final totalmente satisfactorio. Para futuros proyectos, se considerará esta librería con gran prioridad sobre las demás.

**Vagrant:** Lo que Heroku ha conseguido en alojamiento de servidores, Vagrant lo consigue en entorno local. Cuando desarrollamos con muchas dependencias (PHP con Composer, Node.js, PostgreSQL, etc..), instalar y configurar todo el entorno puede ser dificultoso. Las *Vagrant box* permiten comenzar a desarrollar rápidamente sobre un entorno preconfigurado que puede restaurarse a su estado original en cualquier momento. A partir de este proyecto actual, es difícil volver a concebir el desarrollo sin Vagrant.

# Apéndice A

## Lista de entregables

### A.1. Documentos

- El presente documento de memoria de la Práctica Final:  
*PAC\_FINAL\_mem\_Sanchez\_Garcia\_Ruben.pdf*
- El documento de presentación del proyecto:  
*PAC\_FINAL\_prs\_Sanchez\_Garcia\_Ruben.pdf*
- El archivo zip con el proyecto:  
*PAC\_FINAL\_prj\_Sanchez\_Garcia\_Ruben.zip*. Este archivo contiene las siguientes partes
  - El documento con las instrucciones de instalación
  - Un archivo zip con el código
  - Una copia de seguridad de la base de datos
  - Los archivos originales del diseño de las caras del dolor en formato Adobe Illustrator
- El video de presentación del proyecto *PAC\_FINAL\_vid\_Sanchez\_Garcia\_Ruben.mp4*

# Apéndice B

## Código fuente

Debido a su tamaño y la imposibilidad de insertarlo en este documento al mismo tiempo que se mantiene legible, el código se incluye en el archivo adjunto *cronic.zip*

Es posible clonarlo via *git* en <https://git.heroku.com/cronic.git>

La versión instalada se encuentra en <http://cronic.herokuapp.com/>

# Apéndice C

## Capturas de pantalla

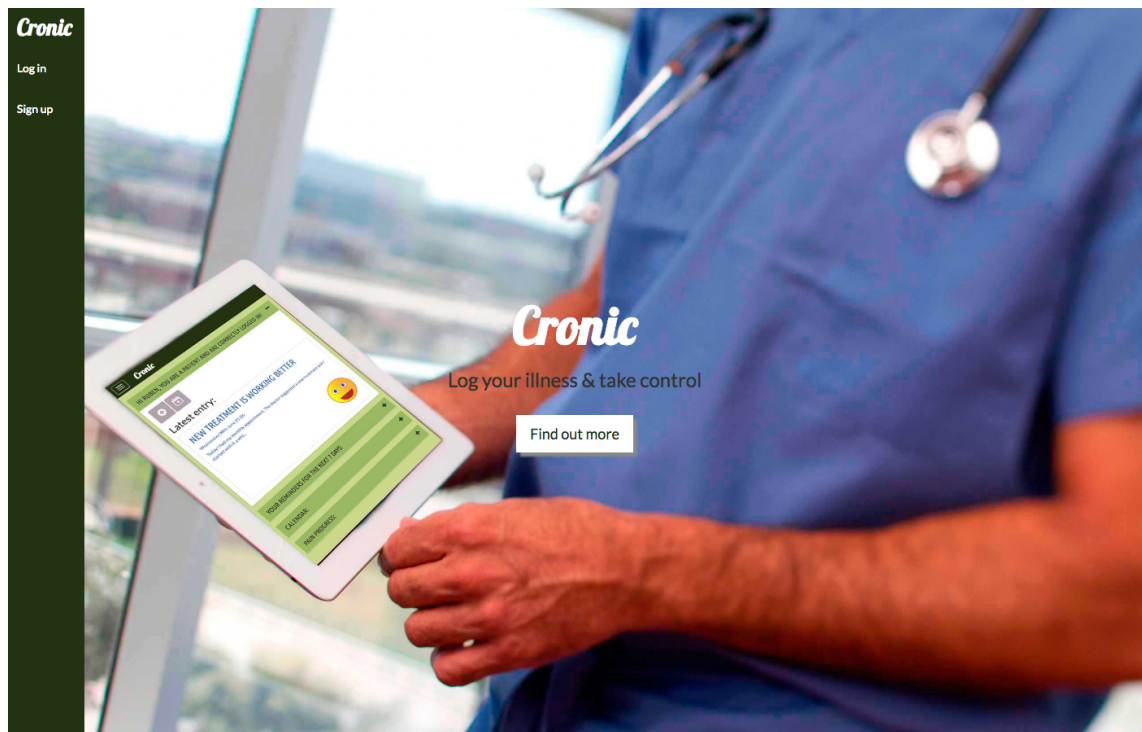


Figura C.1: Portada.

**Cronic**

Log in  
Sign up

### SIGN UP

Name

Surname

E-Mail Address

Password

Confirm Password

Patient or Doctor?  Patient  Doctor

Date of birth:

Height:

Weight:

Allergies:

What is your health problem?

Who is your doctor?

Figura C.2: Formulario de registro

**Cronic**

Log in  
Sign up

### LOGIN

E-Mail Address

Password

Remember Me

[Forgot Your Password?](#)

Figura C.3: Entrada.

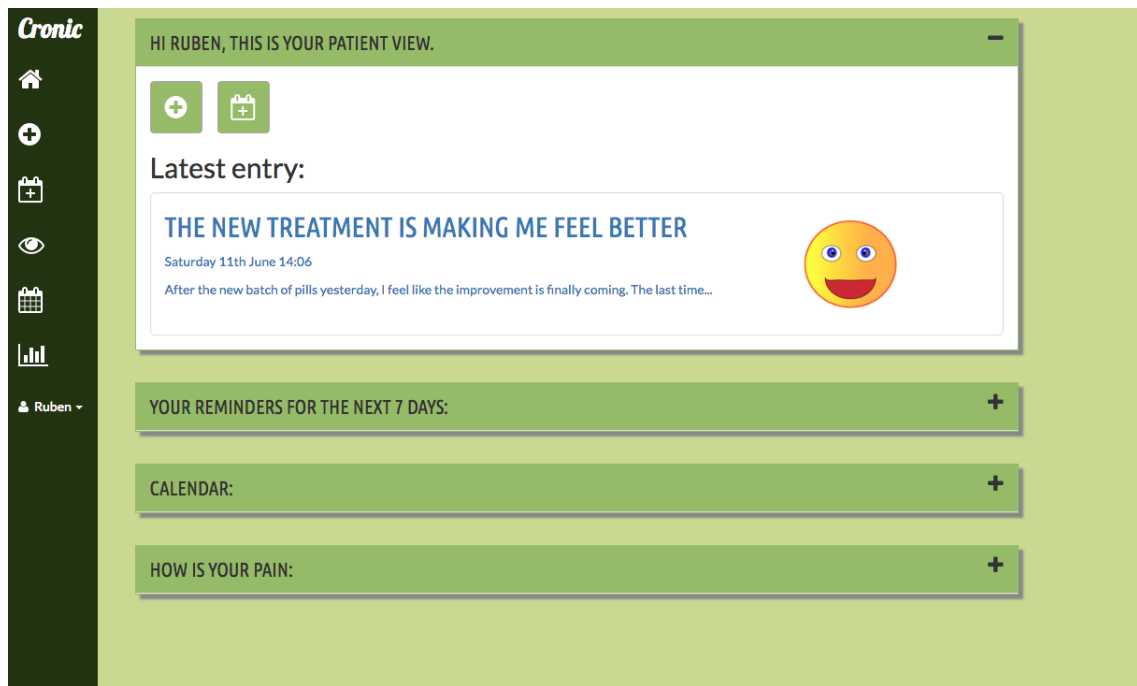


Figura C.4: Panel de control.

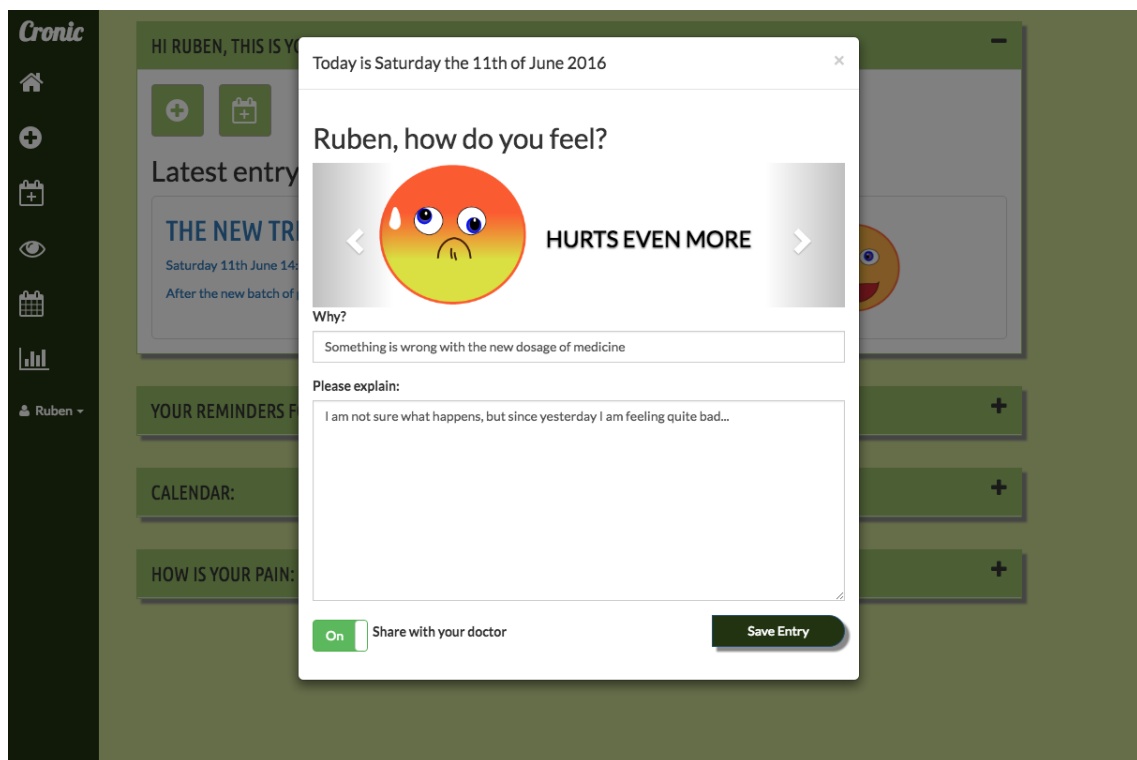


Figura C.5: Introducción de diario.



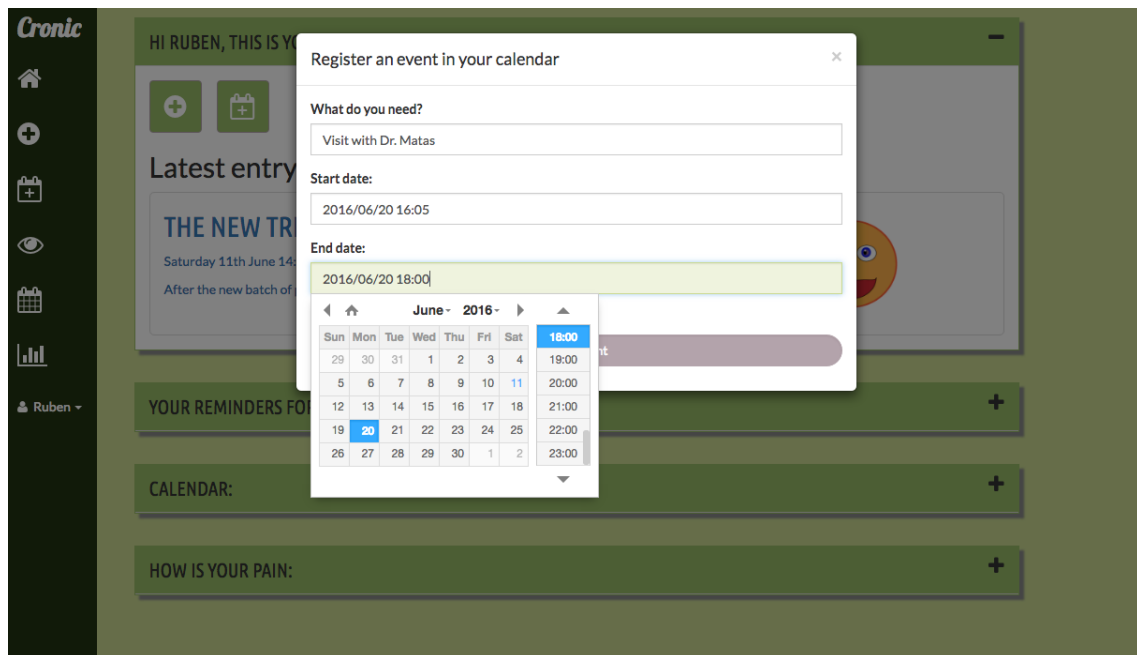


Figura C.6: Introducción de eventos.

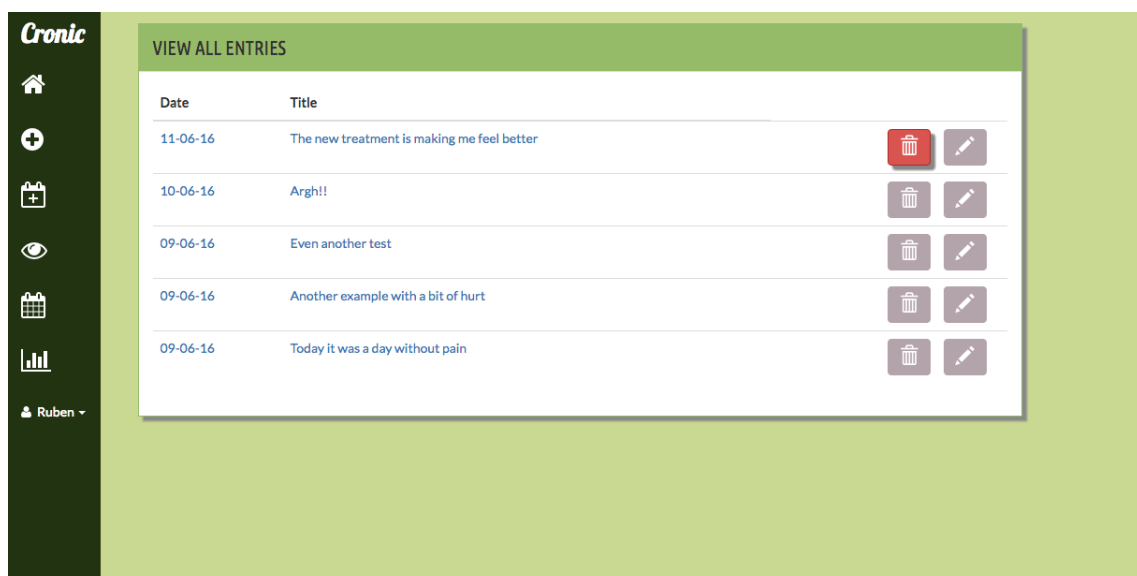


Figura C.7: Listado de entradas.

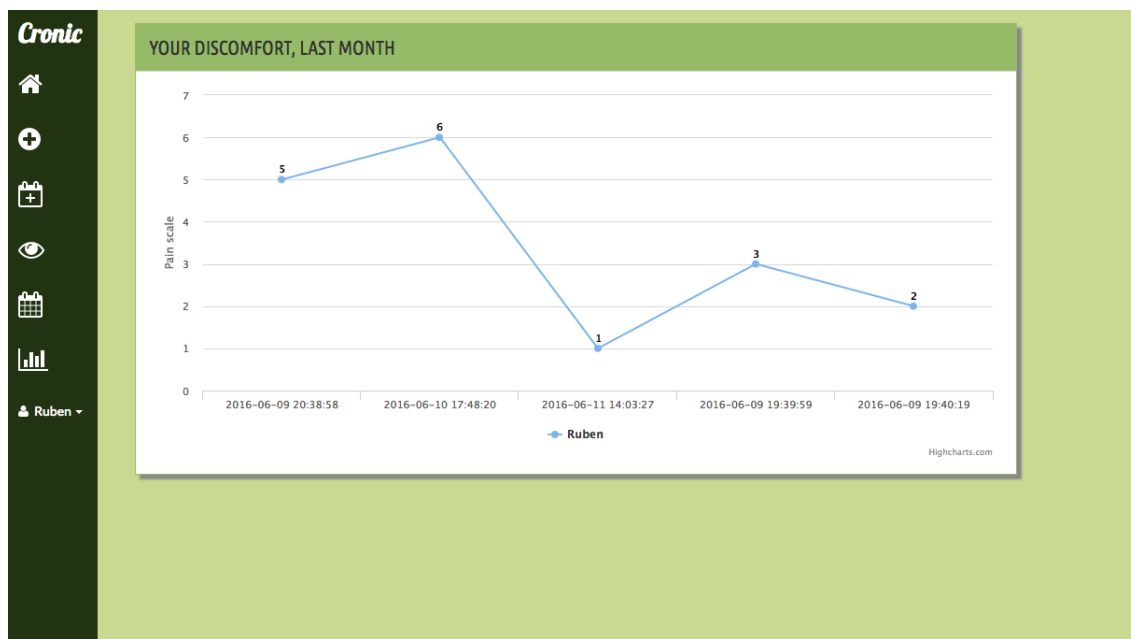


Figura C.8: Gráfico de dolor.

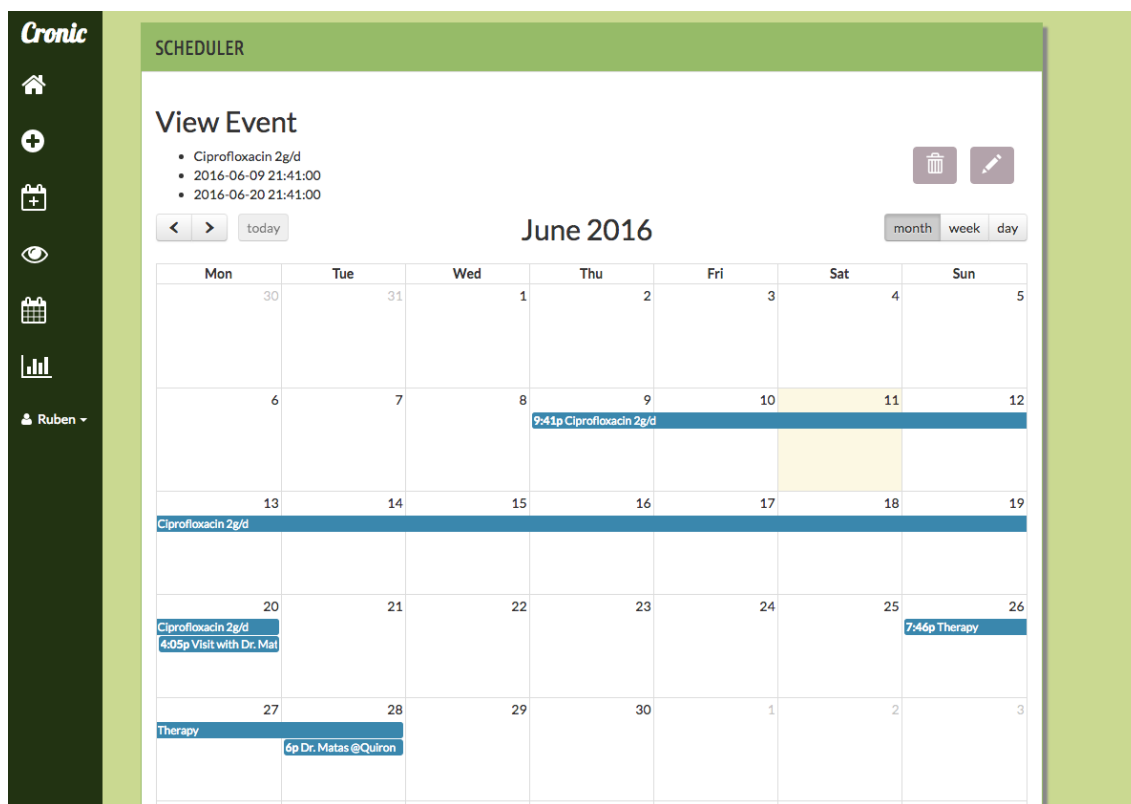


Figura C.9: Vista de calendario.

The screenshot shows the 'Edit your Profile' form in the Cronic application. The form is displayed in a modal window over a dark green background. The form fields are as follows:

- Name:** Ruben
- Surname:** Sanchez
- Date of birth:** 1980-11-22
- Height (cm):** 177
- Weight:** 69
- Allergies:** x Penicillin
- What is your health problem?:** MAV
- Who is your doctor?:** x John Smith

A 'Save' button is located at the bottom right of the form. The background shows a sidebar with navigation icons and a 'YOUR PROFILE' section with an 'Edit your profile' button.

Figura C.10: Edición de perfil.

The screenshot shows the 'YOUR PROFILE' page in the Cronic application after a successful update. The page has a light green background and a sidebar with navigation icons. The main content area displays the following information:

- Settings successfully saved!** (in a green box)
- Hi Ruben, you were born on 1980-11-22
- You measure 177 cm and weight 69 kg.
- Unfortunately you suffer from MAV, under treatment by John Smith.
- Don't forget that your allergy list is Penicillin, so follow your treatment and take care.
- [Edit your profile](#)

Figura C.11: Perfil actualizado.

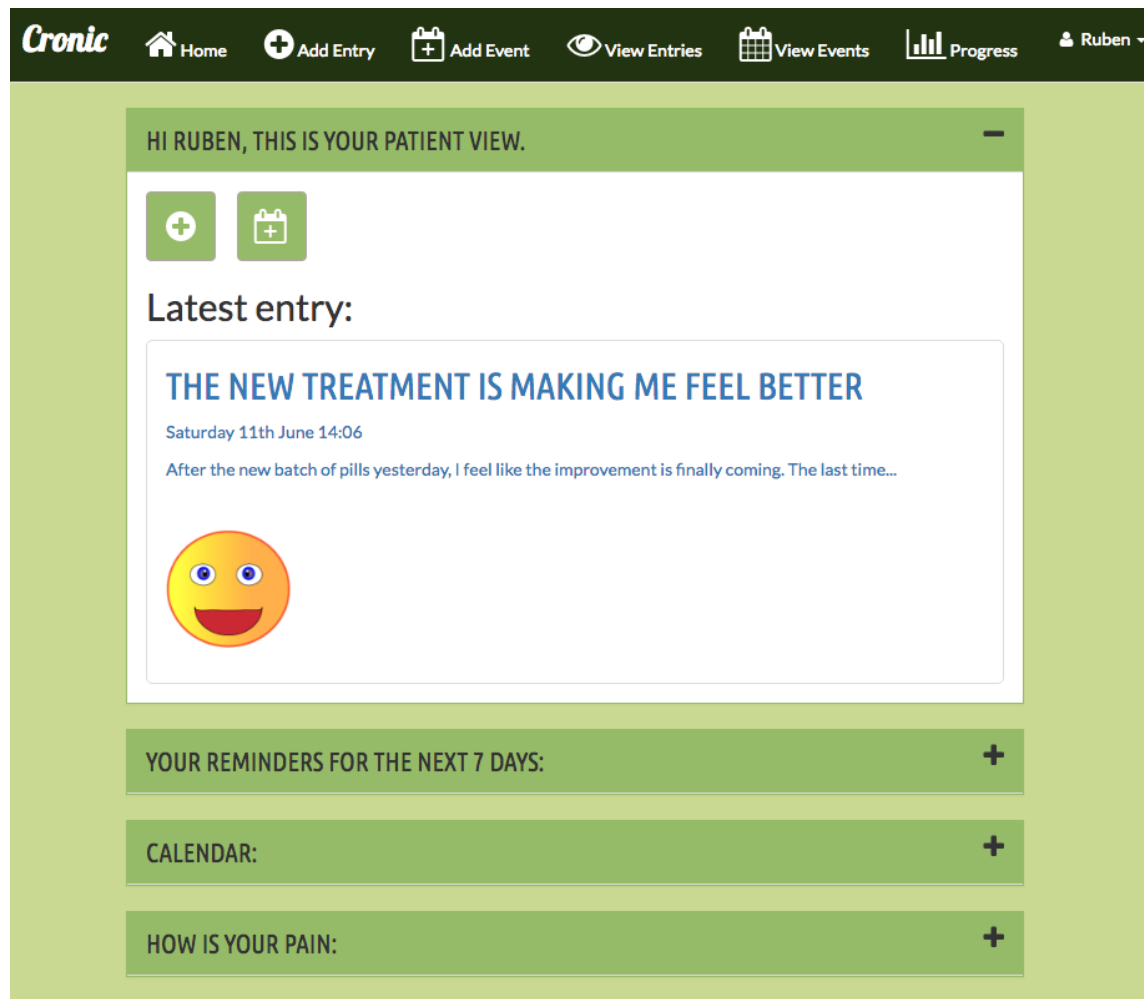


Figura C.12: Distribución de menú en modo tablet.

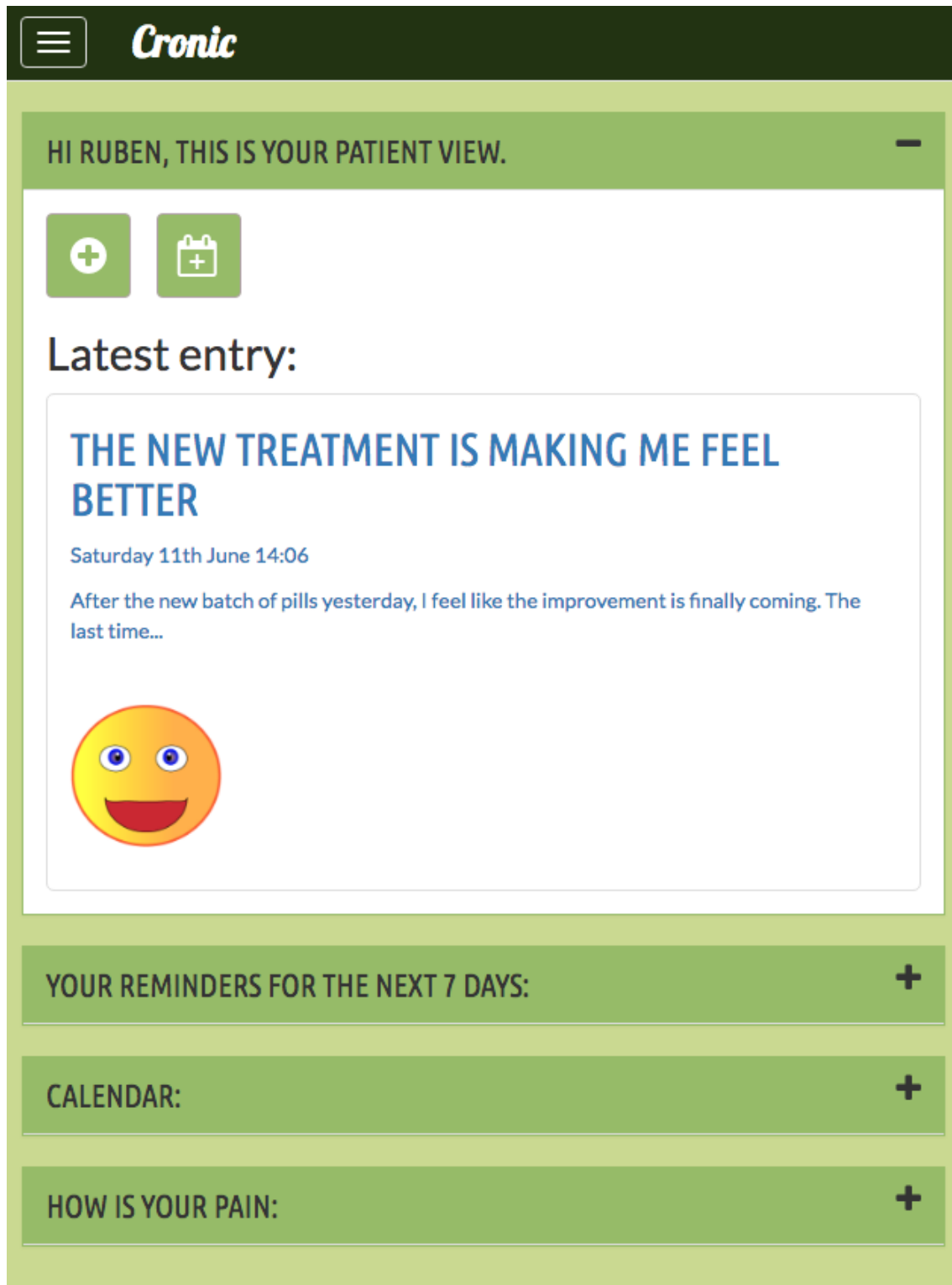


Figura C.13: Distribución de menú en modo teléfono.

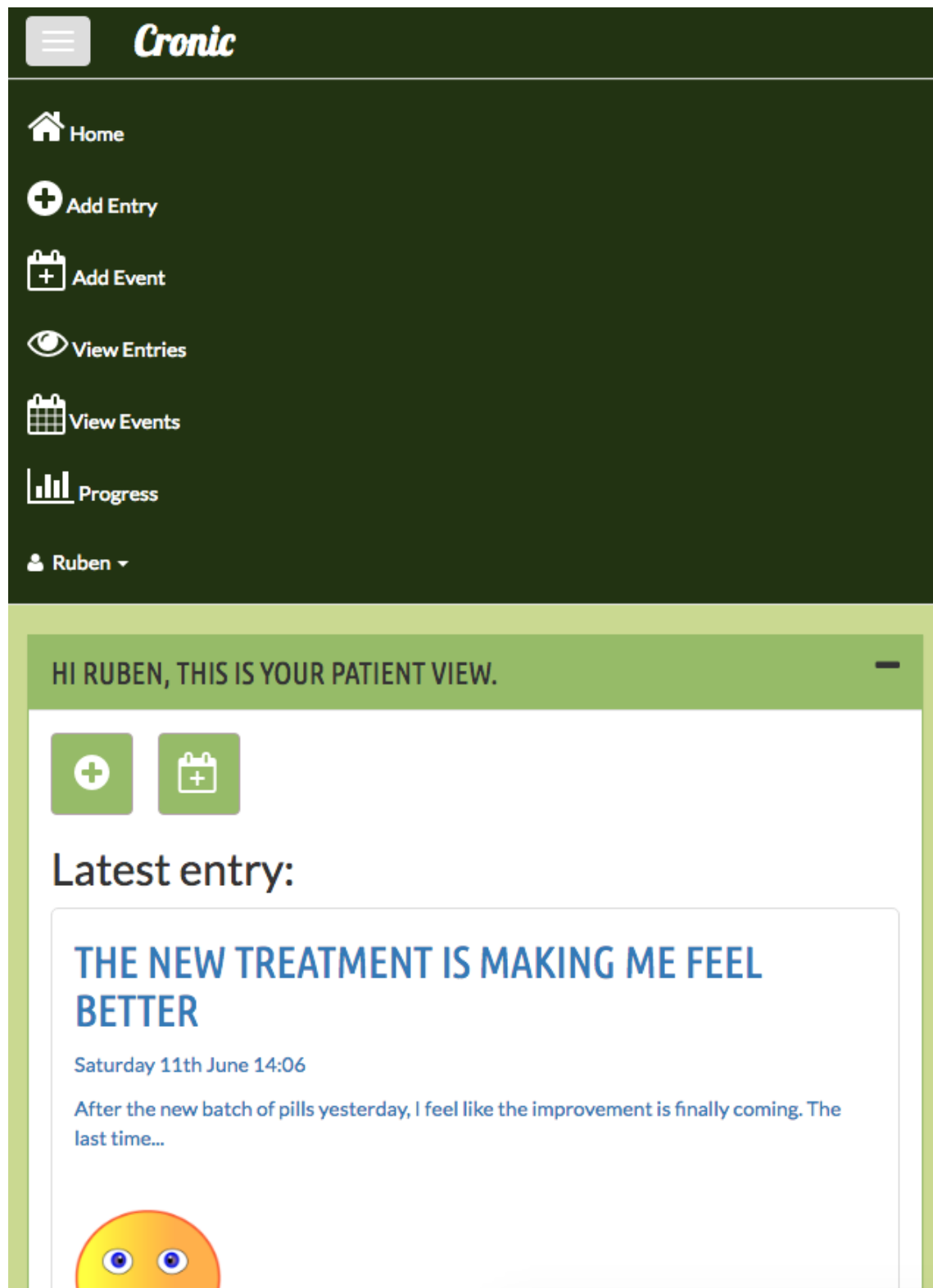


Figura C.14: Menú desplegado en modo teléfono.

# Apéndice D

## Librerías utilizadas

### D.1. Librerías para PHP

- Laravel 5.2

Laravel conforma el grueso de la lógica de la aplicación, ocupándose de gestionar la base de datos a través de *Eloquent*, las vistas, la creación de feeds de *JSON* y las plantillas de texto mediante *Blade*

### D.2. Librerías para Javascript

- jQuery 2.2

*jQuery* se ocupa de gestionar la mayor parte de código ejecutado en el cliente, como por ejemplo las llamadas *AJAX*, las sustituciones de imágenes y las mejoras de interfaz como las opciones de maximizar y minimizar. Asimismo es un requisito obligatorio para el uso de otras librerías utilizadas en la aplicación.

- FullCalendar 2.7

*FullCalendar* toma un feed *JSON* construido con *Laravel* e incluye los eventos detallados en una vista de calendario que requiere *jQuery*.

- Highcharts 4.2.5

*HighCharts* muestra de manera gráfica datos numéricos. En esta aplicación se opta por crear un gráfico con la evolución del dolor del paciente.

- Select2 4.0.2 La herramienta más utilizada para convertir los menús select de html en cajas inteligentes con función de búsqueda

- Datetimepicker 2.5.4 Potente selector de fechas con posibilidad de seleccionar horas.

- Moment.js 2.13.0 Librería para transformar fechas a modo texto

### D.3. Librerías para CSS

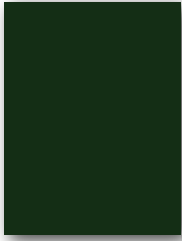
- Bootstrap 3.5.2 Librería de clases CSS para facilitar la creación de estructuras complejas y funcionalidades comunes.
- Select2 for Bootstrap 3.5.2  
Tema para utilizar con nuestros menús *Select2* una estética similar a la de *Bootstrap*
- Bootbox 4.4.0 Complemento para crear popups modales para *Bootstrap*
- Bootstrap toggle 2.2.2 Complemento para sustituir los *checkbox* por interruptores en los formularios.



# Apéndice E

## Guía de estilo

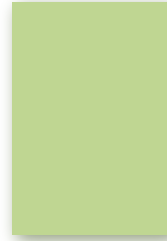
La guía de estilo se adjunta en la página siguiente.



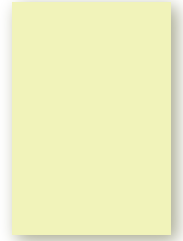
R: 31  
G: 51  
B: 28



R: 150  
G: 187  
B: 104



R: 201  
G: 217  
B: 150



R: 245  
G: 243  
B: 193

*Cronic*

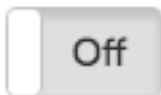
Lobster Regular

TÍTULOS Y CABECERAS

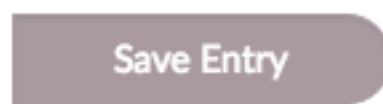
Ubuntu Condensed

Cuerpo de texto

Lato Regular



Bootstrap Toggle Success



Botones hover y normal

Entrada de texto sin foco

Entrada de texto con foco

# Bibliografía

- Dubord, Greg (2011). «Part 9. Thought records». En: *Canadian Family Physician* 57.8, págs. 913-914. URL: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3155448/>.
- Purcell, Maud (2006). «The health benefits of journaling». En: *Psych Central*.
- Shetzer, Lucie (2007). «Confronting Aging and Serious Illness through Journaling: A Study of Writing as Therapy». En: URL: [http://rave.ohiolink.edu/etdc/view?acc\\_num=bgsu1192341678](http://rave.ohiolink.edu/etdc/view?acc_num=bgsu1192341678).
- Smith, Susan y col. (2005). «The effects of journaling for women with newly diagnosed breast cancer». En: *Psycho-Oncology* 14.12, págs. 1075-1082. ISSN: 1099-1611. DOI: 10.1002/pon.912. URL: <http://dx.doi.org/10.1002/pon.912>.
- Taylor, Renée R (2006). *Cognitive behavioral therapy for chronic illness and disability*. Springer Science & Business Media.
- Bieri, Daiva y col. (1990). «The Faces Pain Scale for the self-assessment of the severity of pain experienced by children: development, initial validation, and preliminary investigation for ratio scale properties». En: *Pain* 41.2, págs. 139-150.
- Beck, Kent y col. (2001). «Manifesto for agile software development». En: