Desarrollo de un sistema de control de asistencia de profesorado guiado por pruebas: Test Driven Development

Grado en Ingeniería Informática
Trabajo Fin de Grado
Junio de 2014

Autor Rubén Díaz Martínez
Tutores José Juan Hernández Cabrera. Dpto. Informática y Sistemas
Francisco José Santana Pérez. Dpto. Informática y Sistemas

1. Objetivos

El presente documento detalla el desarrollo de una aplicación para el registro de la asistencia de profesorado a clase.

El proyecto ha sido tutorizado por **José Juan Hernández Cabrera** y **Francisco José Santana Pérez**. En el resto del documento, para abreviar, cuando se haga referencia a José Juan se le nombrará como *tutor* y a Francisco José como *product owner (propietario del producto)*, ya que actualmente ejerce como director de la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria, por lo que, a efectos prácticos, se le considera como tal.

Los objetivos que se persiguen con la realización de este trabajo se pueden separar en dos tipos: a nivel administrativo para el cliente interesado en el desarrollo del producto y a nivel personal/académico para el estudiante.

A nivel administrativo, tenemos los siguientes objetivos clave:

- Como la información estará digitalizada, tratar con ella sería mucho más fácil. En apartados siguientes comentaremos algunos de los trámites que permitirá agilizar esta herramienta.

- A nivel del profesorado, el sistema notificará mediante correos electrónicos si se ha firmado su docencia. Esto sirve a los profesores como registro de la docencia que han impartido, o si les sustituye otro profesor, enterarse de que no ha habido ningún problema.

A nivel personal y académico, se persiguen los siguientes objetivos:

- Participar en el desarrollo de un producto software de principio a fin, algo que no se puede realizar en las distintas asignaturas de la carrera por motivos temporales.

- Aplicar métodos de desarrollo ágil vistos en la carrera pero de un modo más “realista”, al poder contar con un cliente real interesado en el producto que se está desarrollando y del que se puede obtener feedback constante.

- Aplicar el desarrollo guiado por pruebas (Test-Driven Development) en el desarrollo de un software completo, ya que durante la carrera, si bien se explica este proceso y se utiliza en algunas prácticas, solo se usaba para desarrollar características aisladas y no integrarlas en un software.

A continuación, se describe la situación actual del control de asistencia al profesorado en la Escuela de Ingeniería Informática así como las aportaciones del proyecto a dicho proceso. La parte central del documento la ocupa la documentación completa del producto, detallando los requisitos y diseño de la aplicación. Después de esto, se detalla el proceso de desarrollo desde la idea inicial del *product owner*, pasando por la elección de la tecnología para el desarrollo hasta la implementación del producto solicitado. Por último, se sugieren posibles trabajos futuros para mejorar el trabajo desarrollado o nuevas aplicaciones que se apoyen en los datos recogidos y las conclusiones obtenidas, así como las referencias bibliográficas.

2. Estado actual

El desarrollo de este proyecto surge de una idea inicial presentada por el *product owner* como una de sus propuestas electorales para la dirección de la Escuela de Ingeniería Informática (a partir de ahora, **EII**).

Cuando un profesor tiene una clase que impartir, debe firmar para acreditar que ha dado esa docencia. Actualmente en la EII, lo que se hace es poner a disposición de todos los profesores una lista con toda la docencia del día actual y los profesores deben firmarla antes de asistir a clase. Esto es una solución que a nuestro *product owner* no le termina de convencer porque tratar con la información es algo costoso o dado que se puede firmar todo de golpe, habrá profesores que firmen toda la docencia y luego puede surgir un imprevisto que no les permita impartir la docencia que han firmado. En reuniones con el Vicerrector del Profesorado también se comentó la situación actual en otras escuelas universitarias, donde muchas veces ni siquiera se llevaba dicho control a cabo.

La solución que propone implementar el *product owner* es algo muy sencillo. Colocar en un punto del edificio un sistema por el que todos los profesores tengan que pasar antes de ir a clase, identificarse en él, mostrar la docencia que tienen asignada en la hora actual y marcarla como firmada de manera virtual.

La idea de nuestro cliente es desarrollar el proyecto para la EII y, tras un período de prueba, comenzar la implantación en toda la Universidad de Las Palmas de Gran Canaria.

Esta idea surge de un sistema similar que se utiliza actualmente en la UNED, según nos comenta el *product owner*. Cuando un profesor entra en la UNED, debe identificarse en un sistema mediante usuario contraseña para notificar que está en el centro y volver a pasar por ese sistema cuando va a irse. Esto se usa para que los alumnos que vayan a tutoría puedan consultar desde ese mismo sistema si el profesor al que quieren ver se encuentra en el edificio o no.

3. Resultados

A continuación, se describen los resultados obtenidos del desarrollo del sistema. Se detalla la propuesta de valor a la que se ha llegado finalmente, los requisitos y el diseño de la aplicación.

3.1. Propuesta de valor

El valor que aporta este producto se puede ver de dos formas distintas. Desde el punto de vista del *product owner*, el valor que le aportamos es facilitar las tareas administrativas a la hora de consultar datos como pueden ser los siguientes:

- Detectar si una hora de docencia no se imparte.
- En casos de huelga, la administración tiene que comprobar que todos los profesores que no han impartido clases sean aquellos que han hecho huelga. Haciendo una simple consulta al sistema se pueden obtener los profesores que no impartieron clase y comprobar con la lista de los que estaban apuntados a la huelga.

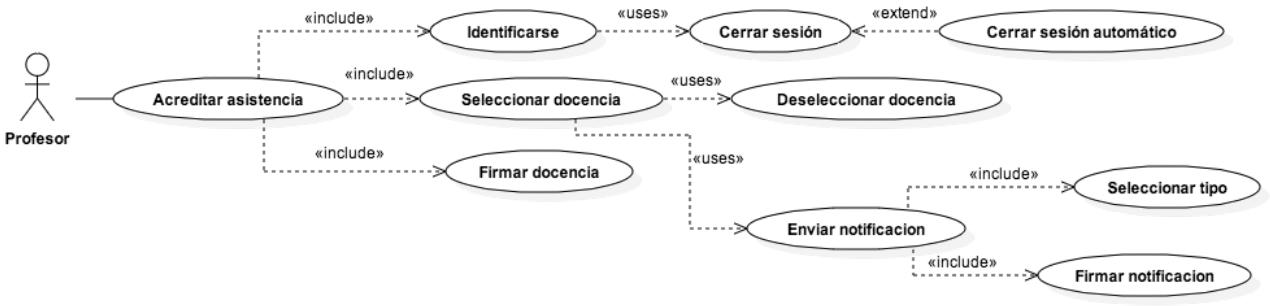
En cambio, para los profesores, el valor lo aportan las notificaciones que mande la aplicación, para tener constancia de que se ha firmado su docencia, ya sea por ellos mismos o por otro profesor.

3.2. Requisitos

En este apartado describimos los requisitos que cumple la aplicación, en forma de casos de uso y la normativa y legislación que nos concierne.

3.2.1. Casos de Uso

Acreditar asistencia



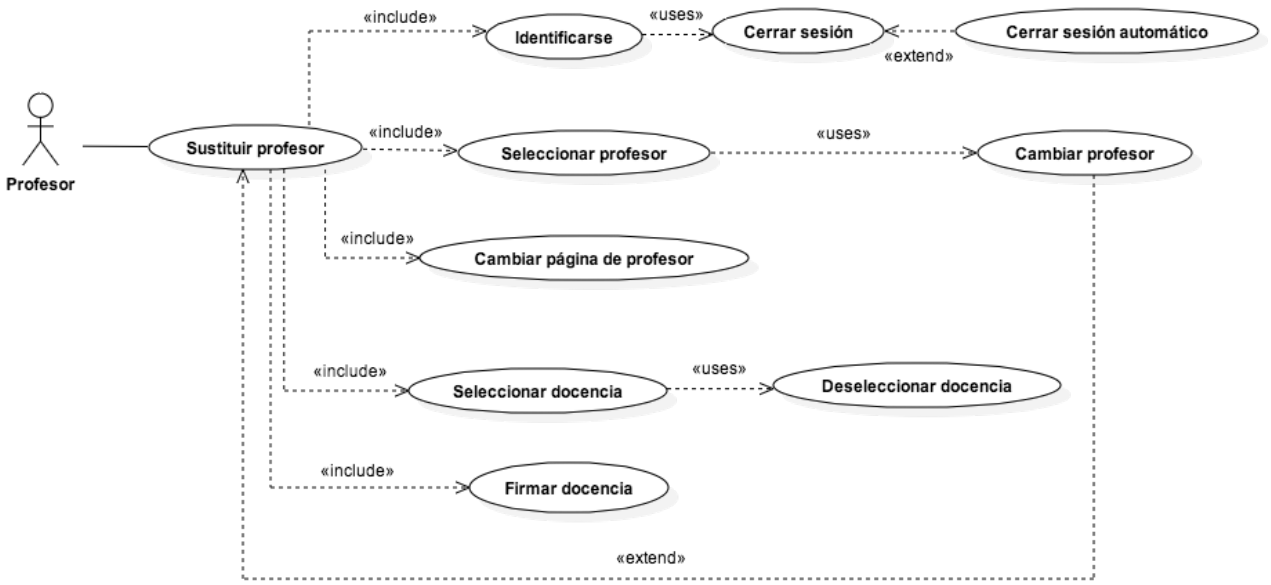
Para que un profesor acredite su asistencia a una clase, debe seguir una serie de pasos sencillos.

- **Identificarse en el sistema** introduciendo su número de DNI. Una vez autentificado en el sistema tendrá la posibilidad de **cerrar sesión** en cualquier momento. Pasado un minuto de inactividad, la sesión se cerrará de forma automática.
- **Seleccionar la docencia** que va a acreditar. Por defecto, se mostrarán seleccionadas las horas de docencia asignadas al profesor en la franja horaria actual, pudiendo **deseleccionarlas** o volverlas a seleccionar. Si el profesor detectase que no se muestra la docencia que debería tener ahora, puede **enviar una notificación**.
  - Para enviar una notificación, el profesor selecciona el tipo de notificación que quiere enviar y firme sobre la tableta e incluye algún detalle si fuera necesario.
- **Firmar la docencia**. Una vez firmada la docencia, se cerrará la sesión. Si se realizara una firma y no se pulsará el botón de confirmación en la tableta, al cerrarse la sesión detectará que hay una firma y guardara la docencia firmada.

Escenario típico:

1. El profesor se identifica introduciendo su DNI.
2. El sistema muestra la docencia que tiene el profesor en la franja horaria.
   2.1. Si no hubiera docencia o ya estuviera firmada, el sistema muestra un mensaje de información.
3. El profesor firma sobre la tableta la docencia que se muestra en pantalla.

Sustituir profesor



El caso de sustituir a un profesor es muy similar:

- La **identificación en el sistema** es exactamente igual que en el caso de *Acreditar asistencia*.
- **Seleccionar profesor a sustituir**. Se muestra una lista con los profesores que tienen docencia en la franja horaria actual. Si la lista de profesores no cabe al completo en la pantalla, se hará una paginación pudiendo el usuario elegir qué página ver. Si el usuario se equivocase de profesor, tendría la posibilidad de cambiar de profesor.
- La **selección de docencia** es igual a *Acreditar asistencia*.
- **Firmar la docencia.**

Escenario típico:

1. El profesor se identifica en el sistema con su DNI.
2. El profesor pulsa el botón "Sustituir profesor".
3. El sistema muestra la lista de profesores con docencia sin firmar en la franja horaria actual.
4. El profesor selecciona el profesor que va a sustituir.
5. El sistema muestra la docencia del profesor seleccionado.
6. El profesor firma sobre la tableta la docencia que se muestra en pantalla.

3.2.2.Normativa y legislación

En el caso concreto de la ULPGC, existe una norma para el control de asistencia a clases teóricas y prácticas. Esta norma contiene 6 artículos, de los cuales nuestra aplicación no cumple todos.

Los artículos 2 y 5 no nos afectan ya que hacen referencia a la obligación de cada centro de tener un horario de docencia y de que los profesores están obligados a firmar dicha docencia.

Nuestro sistema ayuda a cumplir el artículo 1 ya que impone al centro disponer de un medio para el seguimiento y control de la impartición de la docencia. El artículo 6 menciona la obligación de remitir las faltas no justificadas, lo cual también sería fácil contando con los datos que tenemos almacenados.

Los problemas surgen con los artículos 3 y 4. En ellos se dice que la asistencia, tanto a clases teóricas como prácticas debe realizarse mediante la firma en una única hoja por titulación y día, mencionando además que la docencia de práctica debe firmarse en los primeros 15 minutos de la clase. Estos artículos no los cumple el sistema porque precisamente lo que queremos hacer es librarnos de la hoja de asistencia física que dificulta el manejo de datos. Pero es que además el resto de normas que imponen estos artículos no se cumplen, al menos en la EII. En la EII, que es una escuela relativamente pequeña, hay mínimo 4 o 5 hojas de firma por día ya que en una única hoja estándar no caben todas las firmas para la docencia de un día. Además, rara vez la docencia se firma en los primeros 15 minutos de clase (práctica o teórica), ya sea por despistes o por comodidad del profesor.

Dado que la aplicación ha sido solicitada por un director de una escuela universitaria y ha sido del agrado del Vicerrector del Profesorado, no creemos que haya problemas en modificar esta norma para que pueda utilizarse la aplicación en toda la universidad.

Como estamos tratando con datos de carácter personal (DNI del profesor, control de que se encuentra en un cierto lugar a una determinada hora), también tendremos que tener en cuenta la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de carácter personal (LOPD).

En el ámbito concreto de la aplicación que hemos desarrollado para la universidad, en principio no tendríamos que preocuparnos demasiado por esta ley ya que los datos que estamos utilizando serán datos proporcionados por la universidad, la cual entendemos que ya está cumpliendo con la LOPD. Sin embargo, detallamos algunos de los procedimientos clave a fin de informar por si se implantase este sistema en otras empresas.

Según recogen las guías presentes en la web de la Agencia Española de Protección de Datos, el Estatuto de los Trabajadores permite a las empresas (en este caso la universidad) controlar el desarrollo de la prestación laboral. Y dado que estamos

tratando con datos como el DNI y la localización de un profesor a una hora concreta, se aplica en nuestro caso la LOPD, más concretamente el nivel básico.
Lo primero a tener en cuenta es que se debe notificar la creación de ficheros de carácter personal, en nuestro caso, la tabla con los DNI's y nombres de los profesores. Asimismo, tendremos que notificar a los profesores de la existencia de dichero fichero y su utilidad.

A continuación, se debe disponer de un documento de seguridad, que debe contener todas las medidas, normas, procedimientos y reglas para garantizar la seguridad, procedimiento de información al personal, obligaciones del personal y otros apartados descritos en la Guía de Seguridad de Datos de la LOPD.

La guía completa se encuentra en las fuentes de información de esta memoria, pero a modo de resumen citaremos algunos de los contenidos que debe tener el documento de seguridad:

- Indicar las medidas y normas relativas a la identificación y autentificación del personal que acceda a los datos personales.

- Establecer los procedimientos de alta, modificación y baja para el acceso a los datos, incluyendo una lista del personal con acceso.

- Indicar los soportes que contengan datos de carácter personal, los cuales deben estar situados en lugares de acceso restringido y con una lista del personal que tiene acceso a dichos lugares.

3.2. Diseño

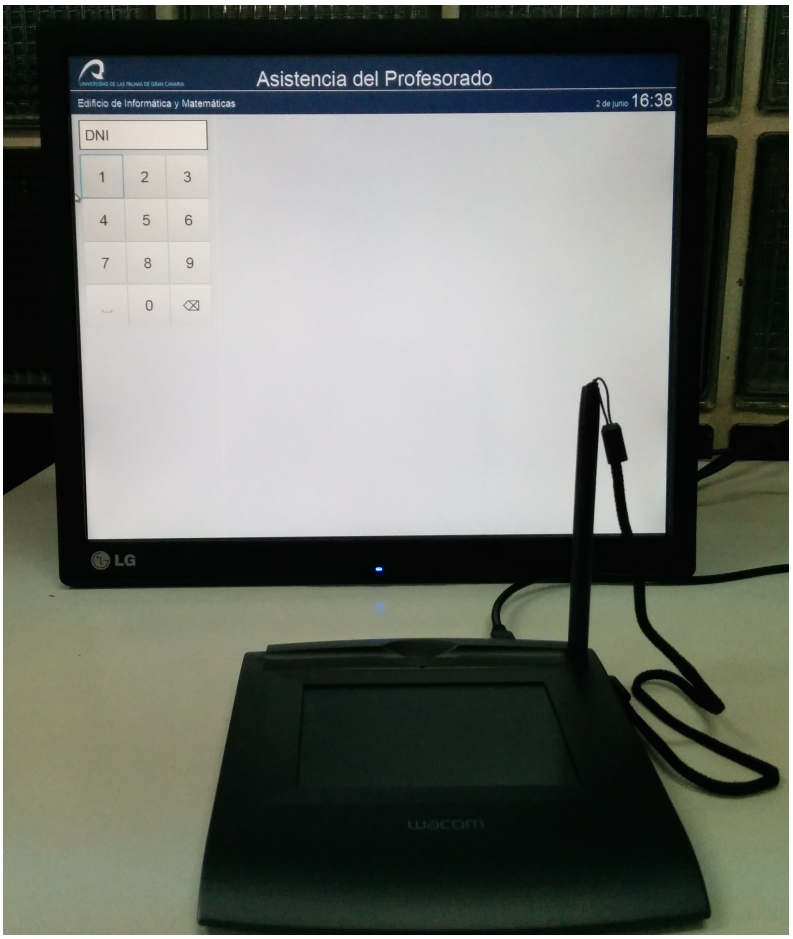
3.2.1. Diseño del sistema

Para poner la aplicación en ejecución, se ha optado por el siguiente hardware:

- Un ordenador pequeño y de características sencillas (1 GB de RAM, 1.8 GHz de procesador), debido a que solo se utilizará para tener durante toda la jornada lectiva la aplicación ejecutándose. La aplicación en sí no impone una restricción del sistema operativo, únicamente la impone la disponibilidad del controlador para la tableta digitalizadora que, en nuestro caso, está disponible para Windows.

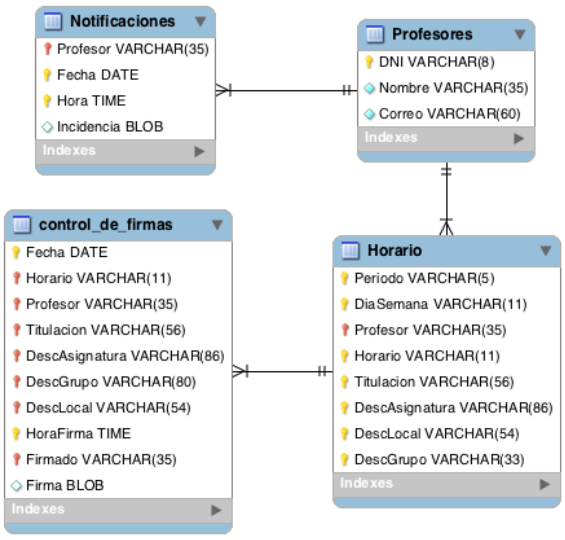
- Un monitor táctil de 17 pulgadas para interactuar con la aplicación, ya que nos parece cómodo y rápido para las acciones que se tienen que realizar.

- Una tableta digitalizadora Wacom, modelo STU 500, el mismo que se utiliza en muchas entidades bancarias, para poder registrar la firma del usuario.



3.2.2. Diseño de la base de datos

El almacenamiento de los datos relativos a la aplicación se hará en una base de datos MySQL cedida por la EII. El diseño de las tablas ha sido muy sencillo ya que nos basamos en las tablas que nos proporciona la universidad para obtener los datos necesarios.

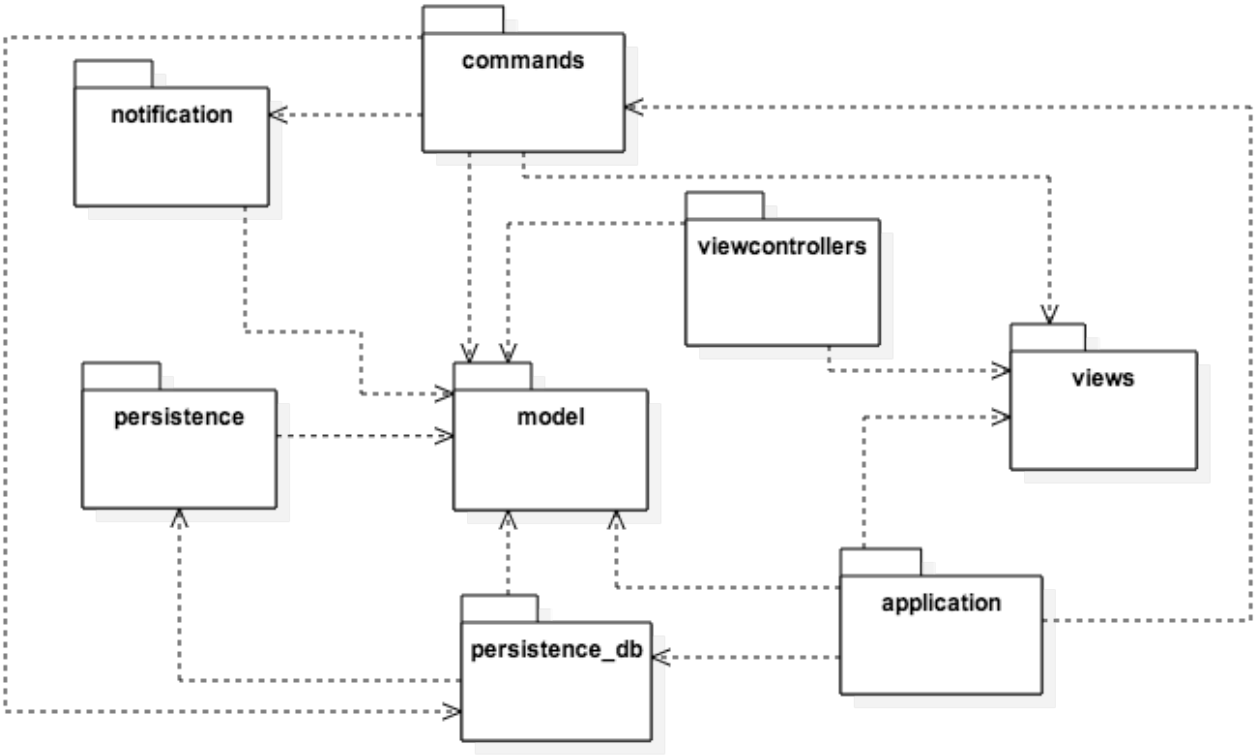


Tenemos una tabla *Profesores* que contiene la información del profesor, es decir, su DNI, su nombre y su correo electrónico.
La tabla *Horario* ha sido adaptada de la base de datos que nos daba la universidad, la cual contiene una entrada para cada hora de docencia de un profesor.
La tabla *control_de_firmas* contiene la docencia que se ha firmado.
La tabla *Notificaciones*, como su propio nombre indica, contiene las notificaciones que han enviado los profesores al usar la aplicación, registrando el profesor que lo envió junto con la fecha y hora.

3.2.3.Diseño del software

Para representar el diseño del software, utilizamos un diagrama de paquetes para ver la estructura general y varios diagramas de clases para ver algunos detalles.

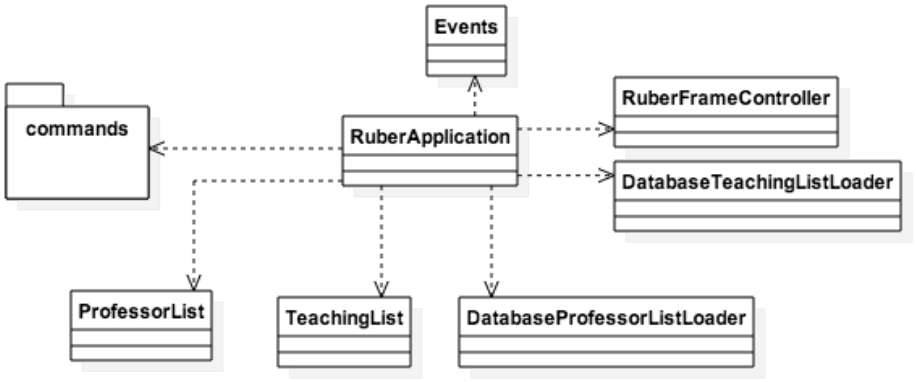
El diagrama de paquetes queda de la siguiente forma, donde podemos apreciar el patrón Model-View-ViewController, donde tenemos las clases de modelo que no dependen de nadie, las de las vistas tampoco y por último los controladores de las vistas que dependen del modelo y de las vistas. Usando JavaFX fue muy fácil separar modelo y vistas, ya que están escritos en lenguajes distintos, el modelo en Java y las vistas en FXML.



- **application**. Contiene la clase principal de la aplicación.
- **commands**. Contiene los comandos que se ejecutan en la aplicación.
- **model**. Engloba las clases que representan el modelo.
- **notification**. Clases relativas a la notificación de la firma de una docencia.
- **persistence**. Interfaces para la persistencia de los datos.
- **persistence_db**. Implementación para la persistencia en base de datos.
- **viewcontrollers**. Controladores de las vistas.
- **views**. Vistas de la aplicación.

A continuación, detallamos el diagrama de clases para cada uno de estos paquetes. Para cada diagrama, se listan aparte las clases pertenecientes al paquete en cuestión y se menciona su función.

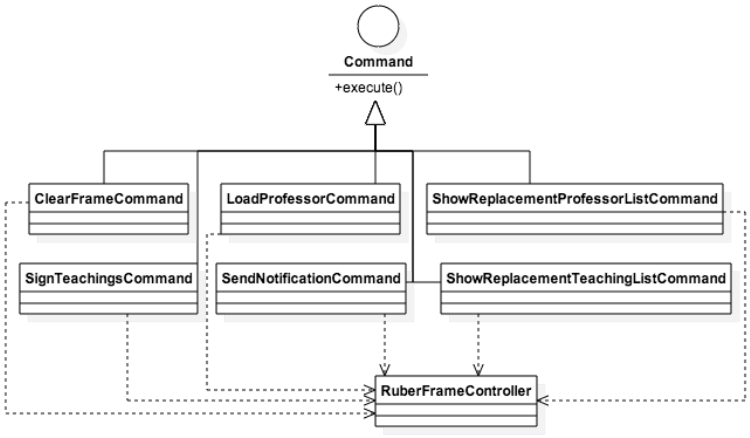
application

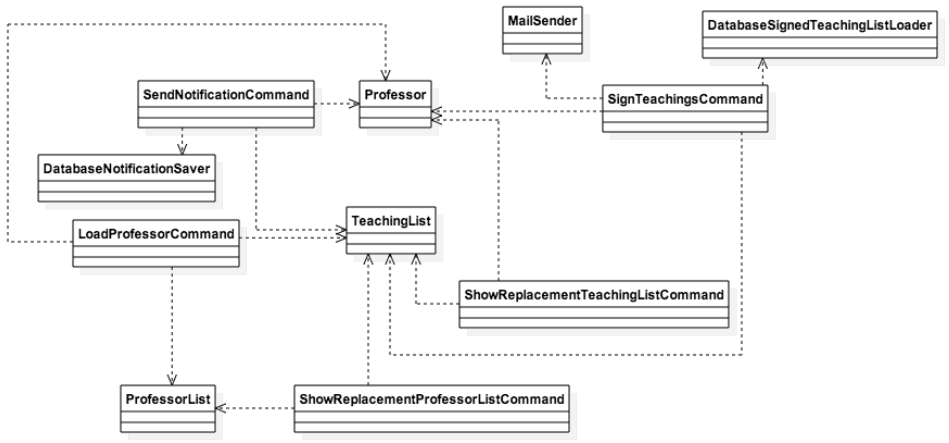


- *RuberApplication.* Es la clase principal de la aplicación. Su función es crear la ventana de la aplicación con toda la información y los comandos necesarios.

commands

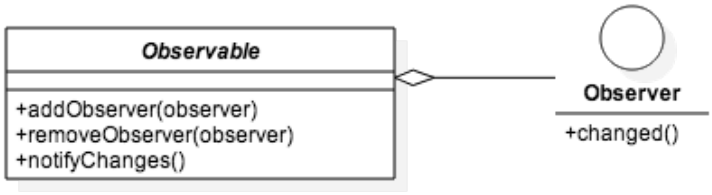
Para mejorar la legibilidad, este diagrama se ha separado en 2. El primero representa la herencia y dependencias comunes a todas las clases, y el segundo las dependencias que no son comunes.



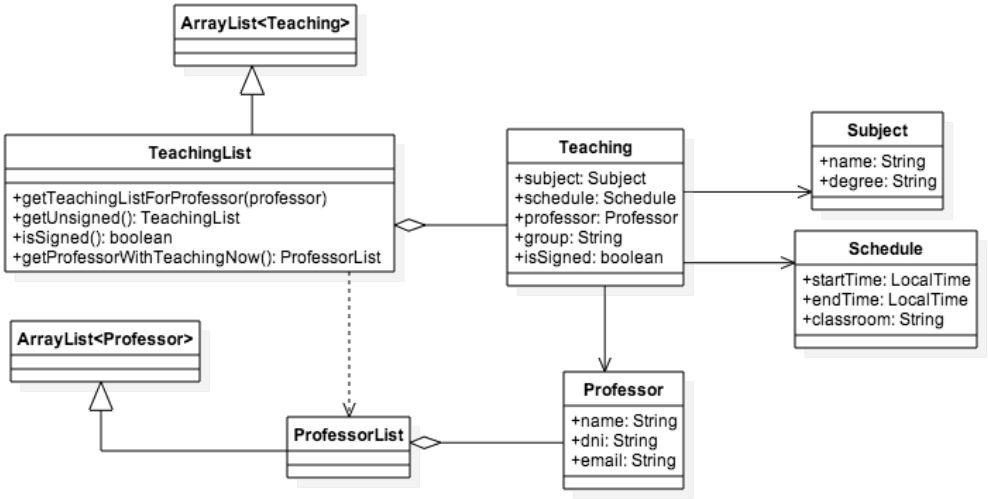


- *ClearFrameCommand*. Comando encargado de reiniciar la ventana de la aplicación al estado inicial, sesión cerrada y mostrando el teclado. Si se hubiese firmado docencia y no se hubiese guardado, se guardaría automáticamente antes de cerrar.
- *LoadProfessorCommand*. Inicia sesión para el profesor asociado al DNI introducido.
- *SendNotificationCommand*. Almacena la notificación que haya mencionado el profesor.
- *ShowReplacementProfessorListCommand*. Muestra los profesores con docencia sin firmar en la franja horaria actual.
- *ShowReplacementTeachingListCommand*. Muestra las asignaturas sin firmar para el profesor al que se va a sustituir.
- *SignTeachingsCommand*. Firma las asignaturas seleccionadas y muestra un mensaje de confirmación.

model

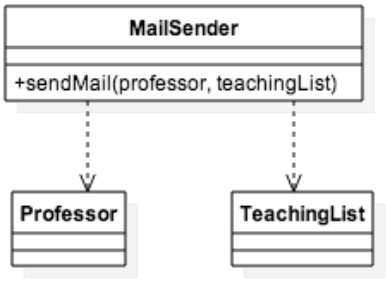


- *Observable* y *Observer*. Clases para implementar el patrón de diseño Observer.



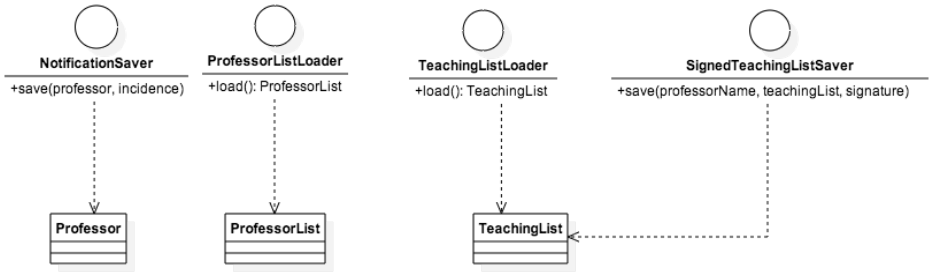
- *Professor*. Representación de un profesor con su DNI, nombre y correo electrónico.
- *ProfessorList*.
- *Schedule*. Horario de la docencia, compuesto por la hora de inicio, fin y el aula.
- *Subject*. Asignatura, compuesta por su nombre y la titulación a la que pertenece.
- *Teaching*. Docencia, la cual se representa mediante un profesor que da una asignatura a un grupo en un horario concreto.
- *TeachingList*.

notification



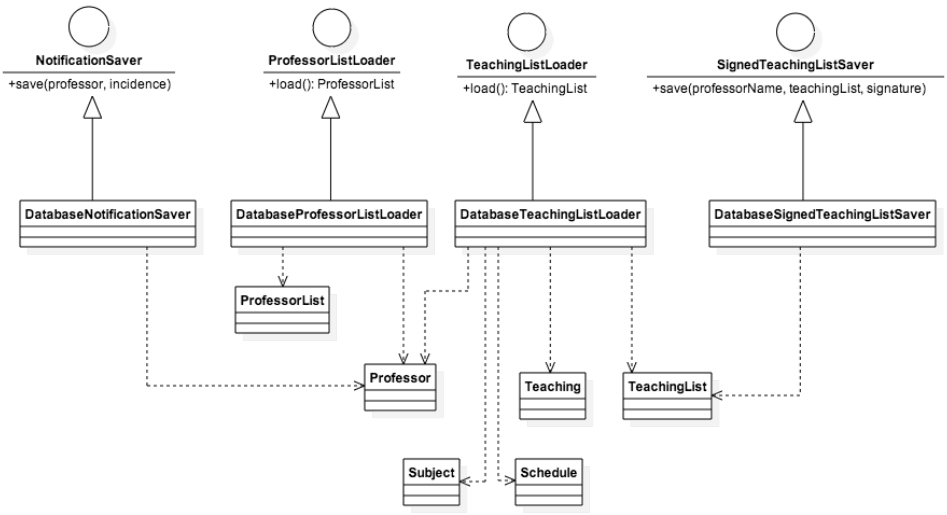
- *MailSender*. Se encarga de enviar un correo al profesor que ha firmado la docencia para confirmar dicha firma. Si hubiese sustituido a otro profesor, también enviará un correo al otro profesor para notificarle que su docencia ha sido cubierta.

persistence



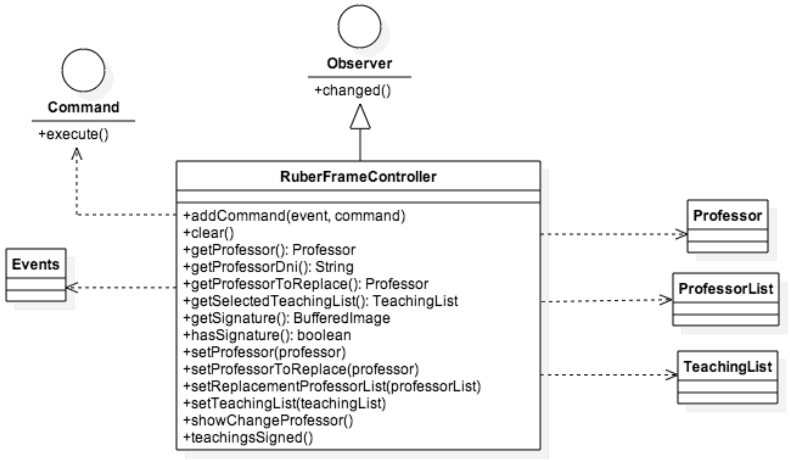
- *NotificationSaver*. Interfaz para guardar una notificación registrada por un profesor.
- *ProfessorListLoader*. Interfaz para representar la carga de toda la lista de profesores mediante un método *load()*.
- *TeachingListLoader*. Interfaz para la carga de la docencia del día actual.
- *SignedTeachingListSaver*. Interfaz para guardar la lista de la docencia firmada por un profesor.

persistence_db



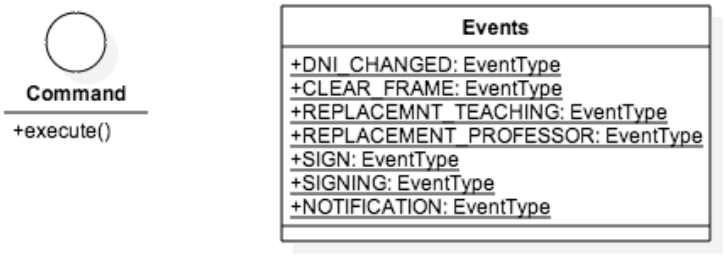
- *DatabaseNotificationSaver*. Implementación para guardar una notificación en una base de datos.
- *DatabaseProfessorListLoader*. Implementación para cargar la lista de profesores desde una base de datos.
- *DatabaseTeachingListLoader*. Implementación para cargar la lista de docencia del día actual desde una base de datos.
- *DatabaseSignedTeachingListSaver*. Implementación para almacenar la docencia firmada por un profesor.

viewcontrollers



- *RuberFrameController*. Controlador principal de la ventana de la aplicación al cual acceden los comandos.

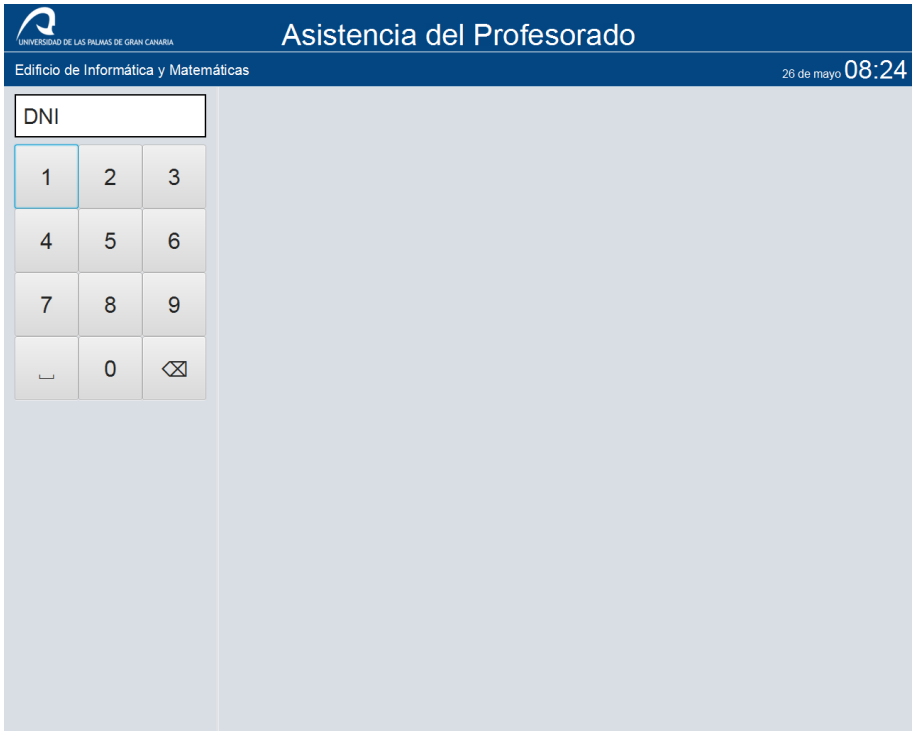
views



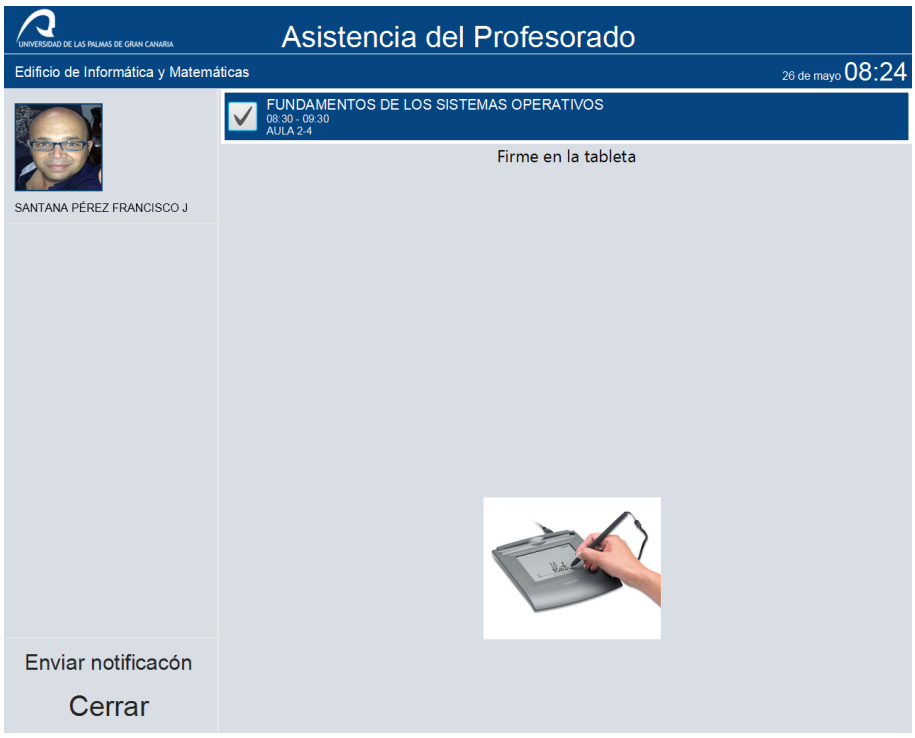
- *Command.* Interfaz que implementan las clases del paquete commands para ejecutar acciones en el sistema.
- *Events*. Contiene todos los tipos de eventos que pueden lanzarse durante la ejecución de la aplicación para que los capturen los controladores.

3.2.4.Diseño de la interfaz de usuario

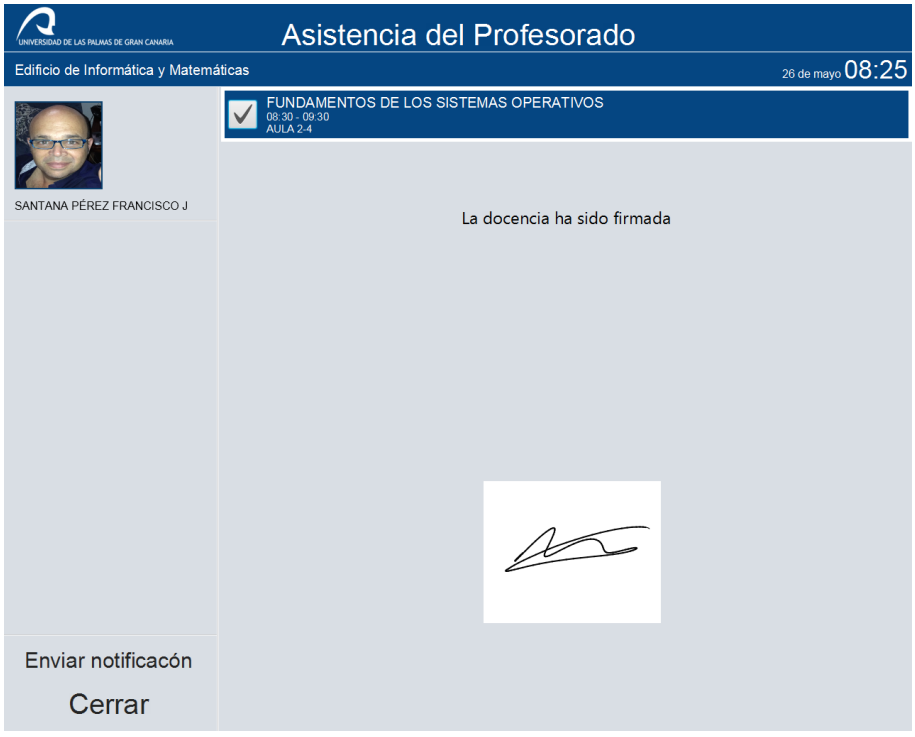
Para mostrar el diseño de la interfaz de usuario, explicaremos el recorrido que habría que seguir para realizar cada uno de los casos de uso. Al iniciar la aplicación, la pantalla inicial es siempre la misma para solicitar el DNI del profesor.



Para el caso de uso de **acreditar asistencia**, el proceso sería el siguiente. Una vez introducido el DNI, se oculta el teclado y se muestra la foto y nombre del profesor, junto con las asignaturas que tiene en la franja horaria actual. Las asignaturas aparecen marcadas por defecto por comodidad, aunque pueden desmarcarse si el profesor no va a asistir a una de las clases.

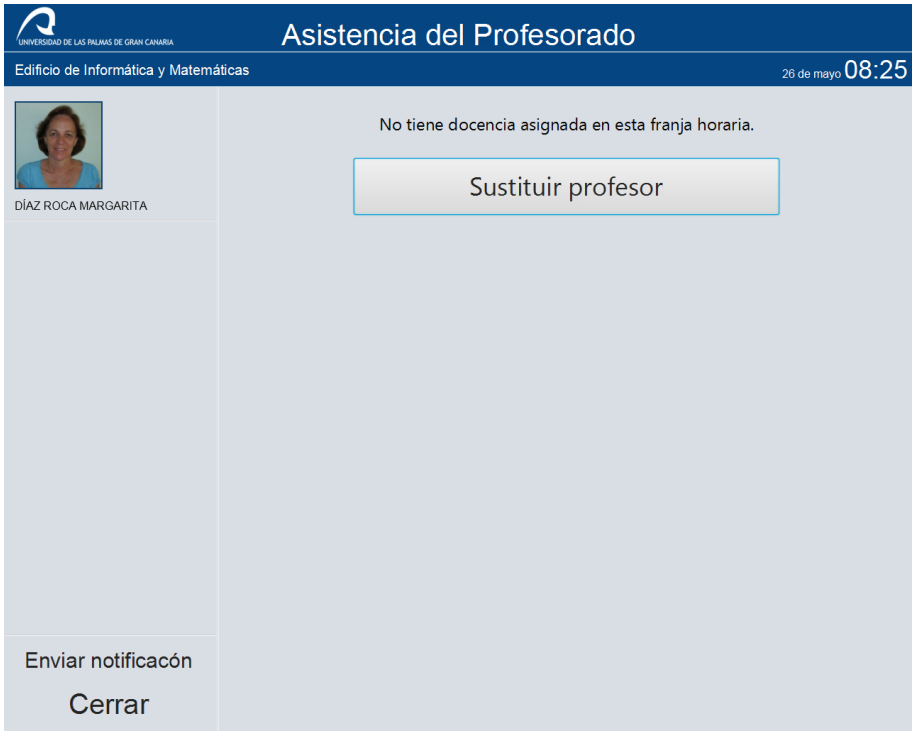


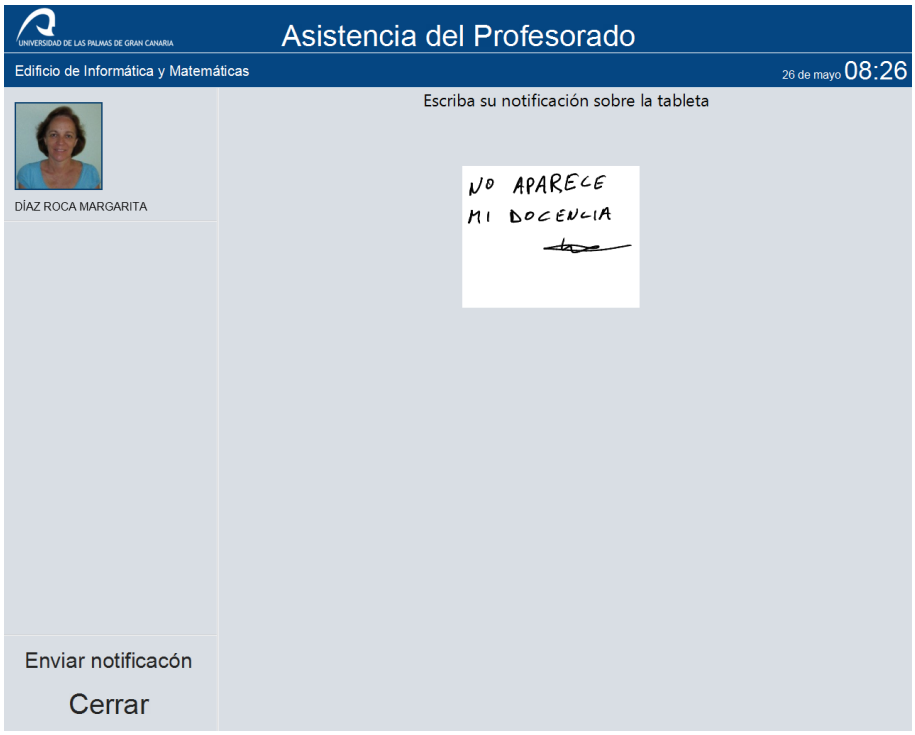
En este momento el profesor debe firmar sobre la tableta digitalizadora y pulsar sobre el botón "Terminar" de la tableta una vez haya acabado de firmar. Cuando haga esto, aparecerá la siguiente pantalla:



Este mensaje aparece durante 3 segundos, tras los cuales se cierra la sesión y volveríamos a la pantalla inicial. Si el profesor firmase y no pulsara el botón "Terminar" en la tableta, cuando se cerrase la sesión (por pulsar el botón "Cerrar" o por inactividad), se firmaría automáticamente.

Si el profesor quisiera enviar una notificación porque su docencia no aparece o un problema parecido, debería pulsar sobre el botón "Enviar notificación":

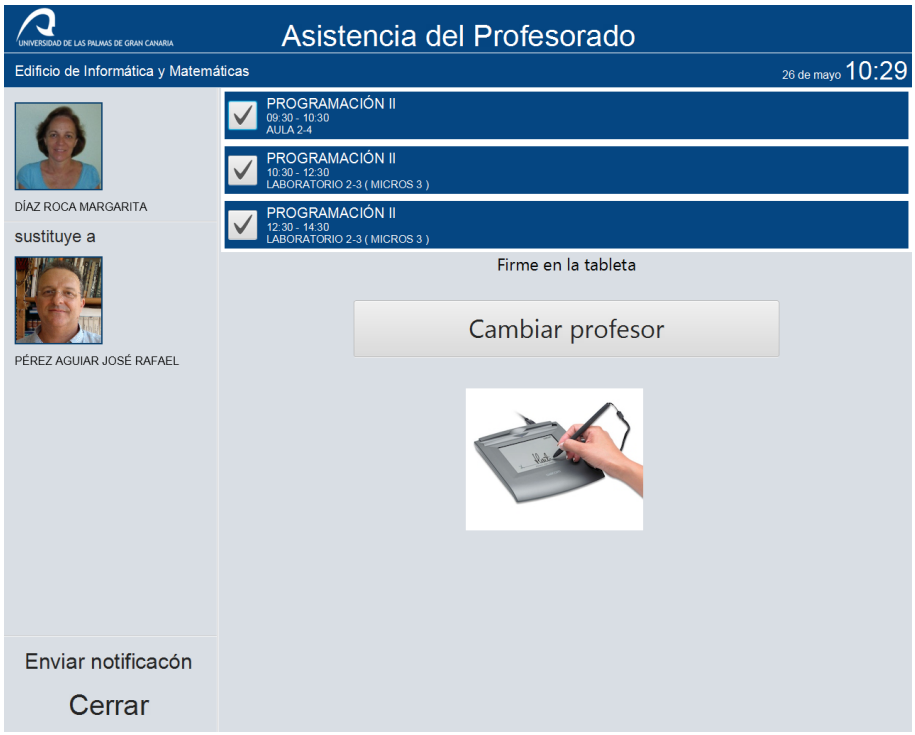




Para el caso de **sustituir a un profesor** no debe tener docencia sin firmar en la franja horaria actual, por lo que al iniciar sesión debería aparecerle esta pantalla.

Al pulsar "Sustituir profesor" verá una lista de los profesores con docencia sin firmar en la franja horaria actual, estando dicha lista paginada por el nombre del profesor, haciendo que la búsqueda sea más rápida.

Una vez seleccionado el profesor, mostraríamos la docencia que tiene dicho profesor sin firmar, dando la posibilidad de cambiar de profesor si nos hubiésemos equivocado.



Por último, firmaríamos en la tableta igual que en el caso de acreditar asistencia.

4. Desarrollo

A continuación se describe el desarrollo seguido, detallando el método utilizado, la tecnología hardware y software y las iteraciones que ha habido hasta llegar al producto final.

4.1.Método de desarrollo

Para el desarrollo del proyecto, se ha optado por seguir métodos ágiles en lugar del modelo clásico o cascada. Esta decisión se ha tomado porque en el modelo clásico se parte de la base de que se pueden definir una serie de requisitos que no van a cambiar a lo largo del desarrollo y a lo largo de los años se ha comprobado que rara vez se cumple.

En contraposición, los métodos ágiles no pretenden definir una serie fija de requisitos, sino aceptar que éstos pueden cambiar y adaptarse al cambio lo antes posible.

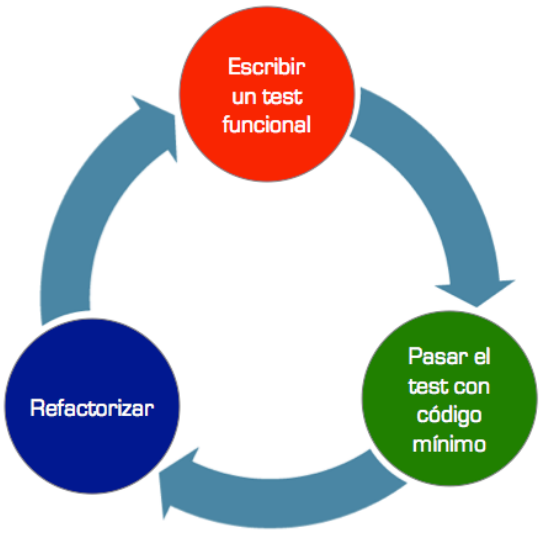
A nivel general de desarrollo, no se seguirá un método ágil concreto (XP, Scrum, Crystal, etc.), sino que se aplicarán algunas de las características de diversas metodologías, como iteraciones cortas tras la que se muestra software funcional al cliente para obtener feedback del producto desarrollado. Se intentarán tener reuniones semanales con el *product owner* para que vea como se va desarrollando el producto y de el visto bueno a las funcionalidades implementadas y al diseño de la interfaz de usuario. Ocasionalmente habrá reuniones con el tutor para resolver dudas o conocer también su opinión en algunas decisiones, ya que el tutor, al ser también un profesor, nos puede dar feedback rápidamente sobre el producto que se está desarrollando.

4.1.1.Desarrollo Guiado por Pruebas

Para el proceso de desarrollo, concretamente lo orientado al "núcleo" de la aplicación (representación del modelo y módulos encargados de la carga o salvado de datos), se ha seguido el desarrollo guiado por pruebas, TDD.

El TDD es un proceso de desarrollo creado por Kent Beck a partir de las prácticas del eXtreme Programming (XP). Consiste en escribir las pruebas automáticas para el código antes de escribir el código. El ciclo de trabajo que se sigue es:

1. Escribir un test que falle. (Red)
2. Hacer el código mínimo para que el test pase. (Green)
3. "Limpiar" el código. (Refactor)



Si buscamos información en internet sobre este método encontramos multitud de ventajas, como:

- Implementamos únicamente las funciones que el cliente necesita.
- Minimiza el número de fallos en el sistema.
- Modulariza el software para que sea altamente reutilizable y preparado para el cambio.
- En palabras de Kent Beck, permite a los programadores dormir tranquilos, ya que después de haber trabajado tienen los tests para comprobar si han roto alguna parte del sistema.

A mi personalmente me parece una técnica perfecta a la hora de empezar a desarrollar un programa, porque en ese momento muchas veces no sabes que hacer primero o vas echando líneas de código sin saber que funcionan hasta que haces un pequeño programa en el *main* de la aplicación o hasta que tienes incluso la interfaz de usuario para probar las cosas.

Además, te permite saber en todo momento qué hacer. Como tu método de trabajo consiste en realizar un test que falle y hacer que el código lo pase, no te ves estresado pensando que tienes que implementar una funcionalidad muy grande ya que vas haciendo test pequeños hasta integrar todo en la gran funcionalidad.

También de cara a iniciar el trabajo día a día supone una ayuda. Kent Beck recomienda que cuando estamos trabajando en solitario al final de una jornada laboral dejemos al menos un test que falle, para así al día siguiente saber por donde empezar, y no perder tiempo mirando el código para ver qué es lo siguiente que implementar.

A la hora de desarrollar, podemos definir 4 tipos de tests:

- Unitarios. Probamos el comportamiento de un módulo aislando todas las dependencias del módulo en cuestión.
- Integración. Consiste en probar varios módulos del sistema para comprobar su correcto funcionamiento.
- Sistema. Se ejecutan simulando al usuario, interactuando con la interfaz de usuario.
- Rendimiento. Comprueban que el sistema funciona bien bajo condiciones extremas y encontrar en qué momento "rompe" la aplicación.

Los más puristas del TDD comentan que todos los tests unitarios deberían ser independientes 100%, pudiendo inyectar todas las dependencias. Sin embargo, bajo mi punto de vista, el concepto de unitario se presta a interpretación. Por ejemplo, si estoy realizando un sistema para hacer operaciones matemáticas, comprobar que una suma se realiza bien para mi es un test unitario, a pesar de que para realizar la suma estemos utilizando 5 clases.

Para los casos en los que queremos hacer un test unitario 100% eliminando todas las dependencias, se utilizan dobles de prueba, que se agrupan en las siguientes categorías:

- Fake. Es una implementación que siempre va a devolver un mismo valor. Por ejemplo, en el caso de la suma, podríamos tener una clase para sumar que siempre devuelva 2.
- Stub. Se utilizan para validar las llamadas a un objeto. En ocasiones a lo mejor nos interesa comprobar que en un método hemos hecho 1 llamada a un objeto, independientemente de si nos ha dado el valor esperado o no. Para esto se usan los stubs, que se implementan con herramientas como Mockito.
- Mock. Son objetos que se implementan con herramientas como Mockito al igual que los stubs. La única diferencia es que aquí estamos comprobando que hemos devuelto el valor esperado. Los mocks se comportan igual que los fakes, con la diferencia de que los fakes son código, en este caso, Java que hemos escrito nosotros y los mocks se crean mediante una herramienta como Mockito.

En la actualidad, el TDD se está consolidando cada vez más en el desarrollo software, llegando a puntos incluso de fanatismo donde se afirma que sin TDD no se puede ser profesional o hacer código limpio. De hecho, este es un tema de bastante actualidad, entre gente de un extremo que aboga porque el TDD es una práctica que más que ayudar daña al desarrollo y gente que como ya comentamos dice que es el único método de desarrollo profesional.

4.1.2.Tests implementados

Para el desarrollo de esta aplicación en concreto, se ha usado TDD para implementar las clases de modelo y persistencia.

Para la clase que representa el horario de la clase, la cual contiene hora de inicio, fin y aula, se implementa un test muy sencillo para comprobar que al pasar ese horario a ristra de caracteres el formato es del estilo "HH:MM - HH:MM".

Para la implementación de la docencia, se usa un test para formatear el nombre de la asignatura. Esto es debido a que de la base de datos el nombre de las asignaturas tienen un estilo tal que "CódigoDeLaAsignatura - NombreDeLaAsignatura (CréditosDeLaAsignatura)". Sin embargo, a la hora de mostrar la asignatura al profesor, solo queremos que vea "NombreDeLaAsignatura". Los otros tests que se implementan son para comprobar la igualdad de dos docencias y para obtener la docencia de un profesor.

Por último, para la lista de profesorado se implementaron dos tests para comprobar que la escritura predictiva devolvía los profesores "candidatos" correctos en cada caso.

Pasando a la persistencia, la implementación de los tests ha sido un poco más problemática. En un principio, se definieron 3 tests para cargar la lista de profesorado, la lista de docencia y firmar una docencia. El problema es que parece lógico pensar que no vamos a usar la base de datos real para hacer estas pruebas, sino que usaremos una base de datos igual pero con datos propios del test. Sin embargo, tampoco nos parece bien que cuando llamamos a la clase encargada de cargar los datos le digamos de qué tabla o base de datos concreta queremos que lea los datos. Hacer que estos tests fueran fáciles de ejecutar implicaría hacer un cambio en la arquitectura de la aplicación que, bajo mi punto de vista, no es beneficioso.

Por este motivo, estas 3 funcionalidades se implementaron haciendo TDD, pero una vez estuvieron validadas con los datos de prueba, se comprobó que funcionaban con datos reales. Pero claro, ya en este caso los tests fallarían. Además, tenemos el problema de que, la carga de docencia por ejemplo, es condicional al día en el que se ejecutan ya que cargará una docencia u otra.

4.2.Tecnología

Al iniciar el proyecto, la idea era realizar una página web desde la que los profesores se identificarían con MiULPGC para firmar la docencia que iban a impartir. Por este motivo, empezamos utilizando como tecnología software Java como lenguaje en el servidor y JavaScript con la librería jQuery como lenguaje para el cliente, y como frameworks de prueba para realizar TDD, JUnit para Java y QUnit para JavaScript. Además, tener una solución web nos daba la facilidad de que sí en un futuro queríamos permitir a los profesores firmar su docencia desde un teléfono móvil (tal vez usando geolocalización para comprobar que no firman desde casa), no habría que hacer demasiados cambios a la aplicación.

A las pocas semanas el tutor sugirió utilizar el framework GWT de Google, el cual permite escribir únicamente código Java y compilarlo a HTML, CSS y JavaScript. Siguiendo este consejo, se empezó a utilizar este framework con el nunca había trabajado.

La experiencia con GWT fue bastante buena, ya que como todo era Java, al hacer TDD solo dependía de JUnit y no tenía que ejecutar pruebas en el entorno de desarrollo y otras en el navegador, como tenía que hacer con QUnit. Además, las interfaces de usuario con GWT pueden hacerse en ficheros XML, lo cual, desde mi punto de vista, es bastante cómodo y también se hace en otros entornos como es en Android.

Sin embargo, GWT tenía una serie de inconvenientes. Con la reciente salida de Java 8, quería poder utilizarlo en el proyecto y sacar partido a las nuevas características del lenguaje, como las expresiones lambda. Sin embargo, el framework GWT solo soporta, hasta el momento, Java 6. Además, ya que a fin de cuentas estamos trabajando con código Java en lugar de HTML, un cambio insignificante en la interfaz que se actualizaría de forma automática en HTML, en GWT suponía casi 1 minuto de compilación en mi equipo de trabajo.

Comentamos estos problemas con el tutor y convenimos en cambiar de tecnología y utilizar JavaFX. En lugar de generar una página web haremos una aplicación de escritorio, ya que el hecho de firmar de manera física con una tableta impide dar la posibilidad de firmar desde distintos dispositivos electrónicos. Además, la ventaja de JavaFX es que nos permite conservar la declaración de interfaces de usuario en ficheros con formato XML.

Con respecto al almacenamiento de los datos, la base de datos que se va a utilizar es MySQL, principalmente porque es la que nos ofrece la EII en su servidor.

Pasando a la tecnología hardware, para ejecutar la aplicación se utiliza un ordenador muy básico (1 GB de memoria RAM, 1.8 GHz de procesador) con Windows, junto con un monitor táctil de 17 pulgadas y un tableta Wacom STU 500 para capturar la firma.

4.3. Iteraciones

El desarrollo del producto estará basado en iteraciones, con reuniones frecuentes con el cliente para recibir feedback de manera rápida.

La primera iteración estará dedicada a implementar las funcionalidades que pide el *product owner*, es decir, los casos de uso detallados más arriba. Una vez validadas las funcionalidades, se hará una segunda iteración para mejorar aspectos de la interfaz de usuario. Cuando ésta haya acabado, se pondrá en funcionamiento la aplicación con algunos profesores para obtener feedback de los usuarios. Por último, se hará una tercera iteración para mejorar lo que hayan comentado los usuarios y algunos cambios en al interfaz gráfica.

4.3.1. Idea inicial

La idea inicial del producto es que sea una página web a la que se accederá desde un equipo con pantalla táctil en la consejería de la EII. Desde ahí, los profesores se autentificarán con su usuario y contraseña de MiULPGC. Una vez autentificado, se muestra al profesor su próxima docencia. Este era el concepto que tenía el *product owner*:

Con esto en mente, comenzamos a buscar información sobre cómo obtener todos los datos necesarios a través de la ULPGC, es decir, horarios de docencia y un servicio para autentificar al profesorado.

Para obtener los horarios de docencia, nos pusimos en contacto con el Servicio de Informática de la ULPGC, el cual nos comunicó que no existía aplicación o servicio para obtener esos datos, por lo que nos los debería proporcionar el director de la EII. Contactamos con el *product owner*, que nos proporciona un fichero Excel con todos los horarios de docencia que luego importamos a una base de datos MySQL proporcionada por la EII.

Para la autentificación de profesorado el cambio tuvo que ser un poco más radical. Hace pocos meses la ULPGC implantó un nuevo sistema de autentificación, CAS (Central Authentication System), mediante el cual todos los servicios de la universidad que requieran el uso del usuario y contraseña de MiULPGC deben pasar por esa web para autentificarse. Anteriormente existía un servicio de autenticación, pero con este nuevo sistema todas las aplicaciones deben pasar por la web del CAS y trabajar con "tickets" en lugar de usuario/contraseña.

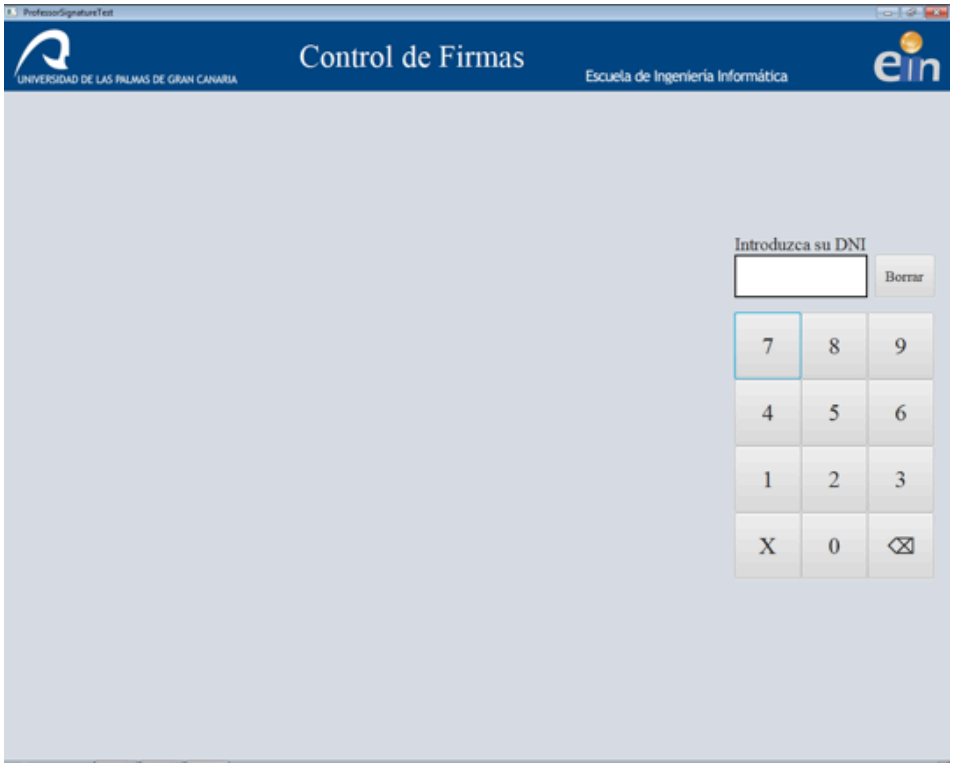
Este cambio supone una problemática porque, dado que la aplicación que vamos a desarrollar esta pensada para que se ejecute en un ordenador compartido por todo el profesorado, estaríamos continuamente iniciando y cerrando sesión con el CAS, algo que no es lo que queremos. Únicamente necesitamos validar un usuario, no iniciar una sesión para luego cerrarla a los 10 segundos. Además, en conversaciones posteriores con el *product owner*, llegamos a la conclusión de que introducir diariamente varias veces el usuario y contraseña de MiULPGC en un monitor táctil es un proceso engorroso y peligroso para que otras personas descubran la contraseña.

Con esto en mente, proponemos al *product owner* otros métodos para autentificar a los profesores. La idea de tener un PIN para cada profesor no gusta porque al ser una contraseña única para este sistema es más fácil que los profesores la compartan y firmen en nombre de otros. Otra idea planteada fue autentificar mediante una foto, al llegar al monitor se hace una foto y se registra quien firmó la asistencia. Al *product owner* tampoco le convenció está opción y al final se optó por utilizar una tableta digitalizadora, con la que los profesores firmen la asistencia como si fuera en papel, pero almacenando todo de manera digital.

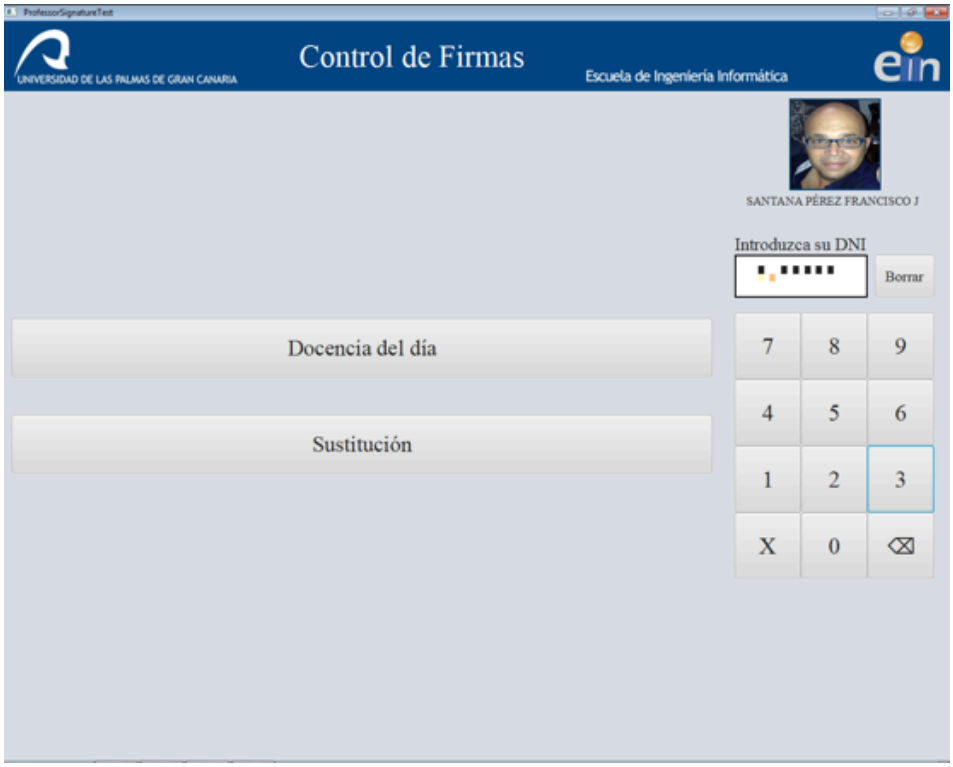
4.3.2. Iteración 1. Funcionalidades

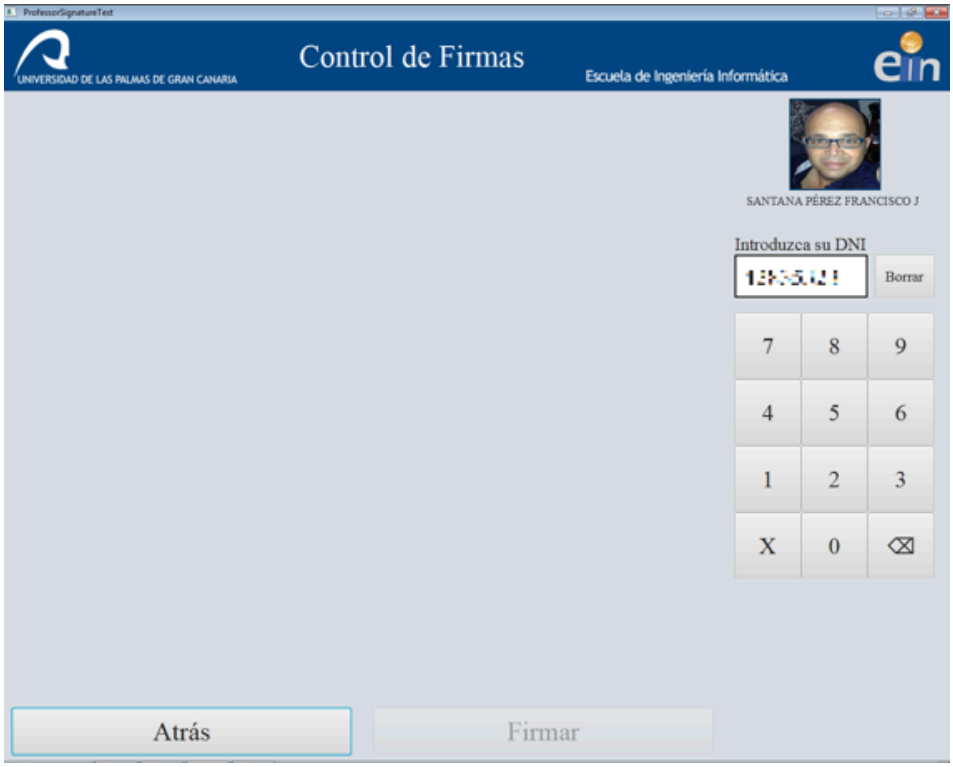
Una vez a la semana, aproximadamente, se tenía una reunión con el cliente para validar lo que se iba implementando. De esta forma se iban haciendo cambios muy sencillos sobre la marcha, como por ejemplo el color de alguna parte de la interfaz o que se permitieran firmar asignaturas dentro de un rango de tiempo determinado. Si se hubiese seguido un enfoque de desarrollo en cascada, el cliente hubiese visto el producto terminado al final y probablemente se acumularían los cambios a realizar y no serían tan sencillos.

Tras las reuniones con el *product owner*, se llegó a un primer prototipo funcional que tenía la siguiente interfaz. Al arrancar el programa se mostraba esta pantalla que solicitaba el DNI. El teclado se hizo similar al teclado numérico de un ordenador, añadiendo una X para los DNI de extranjeros.

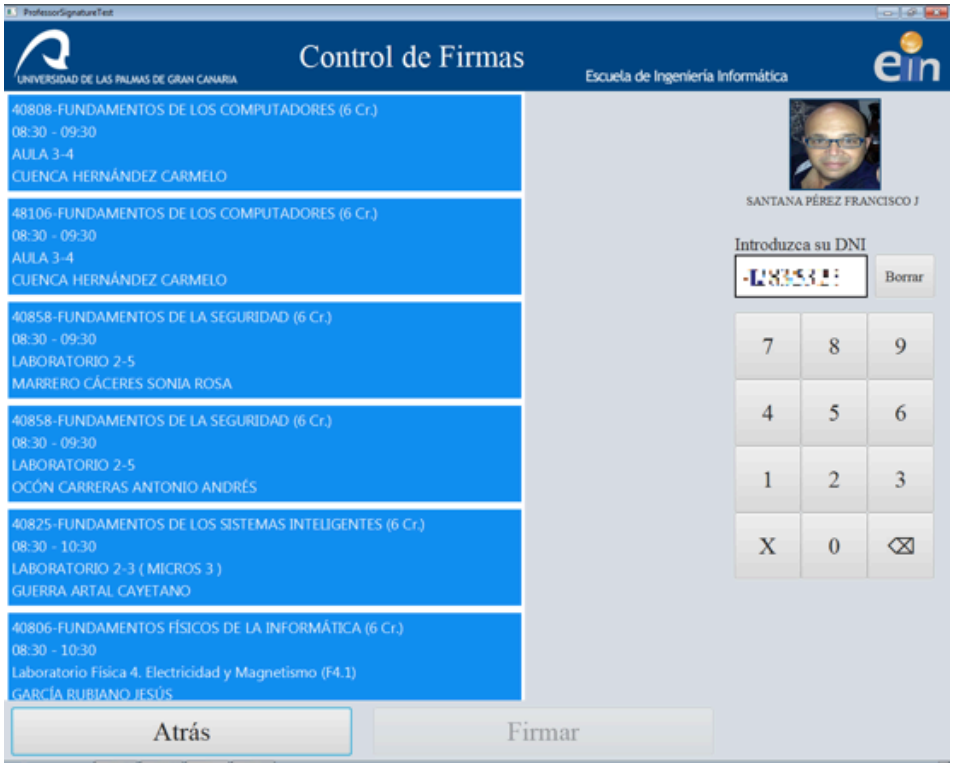


Tras introducir el DNI, se muestra el nombre e imagen de perfil del profesor, junto con las dos opciones disponibles, ver la docencia del día o sustituir. En este caso, si pulsamos “Docencia del día” nos aparece una pantalla vacía porque no tenemos docencia y nos da la opción de volver atrás.

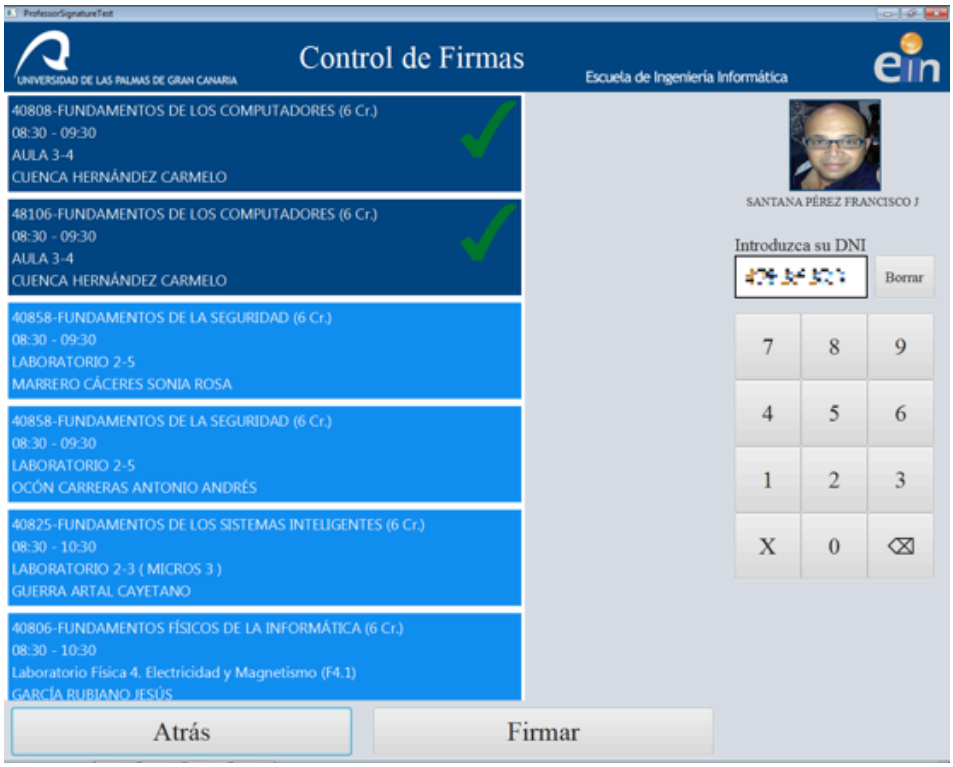

ProfessorSignatureTest
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Control de Firmas
Escuela de Ingeniería Informática
ein
SANTANA PÉREZ FRANCISCO J
Introduzca su DNI
Borrar
Docencia del día
Sustitución
7
8
9
4
5
6
1
2
3
X
0


ProfessorSignatureTest
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Control de Firmas
Escuela de Ingeniería Informática
ein
SANTANA PÉREZ FRANCISCO J
Introduzca su DNI
Borrar
7
8
9
4
5
6
1
2
3
X
0
Atrás
Firmar

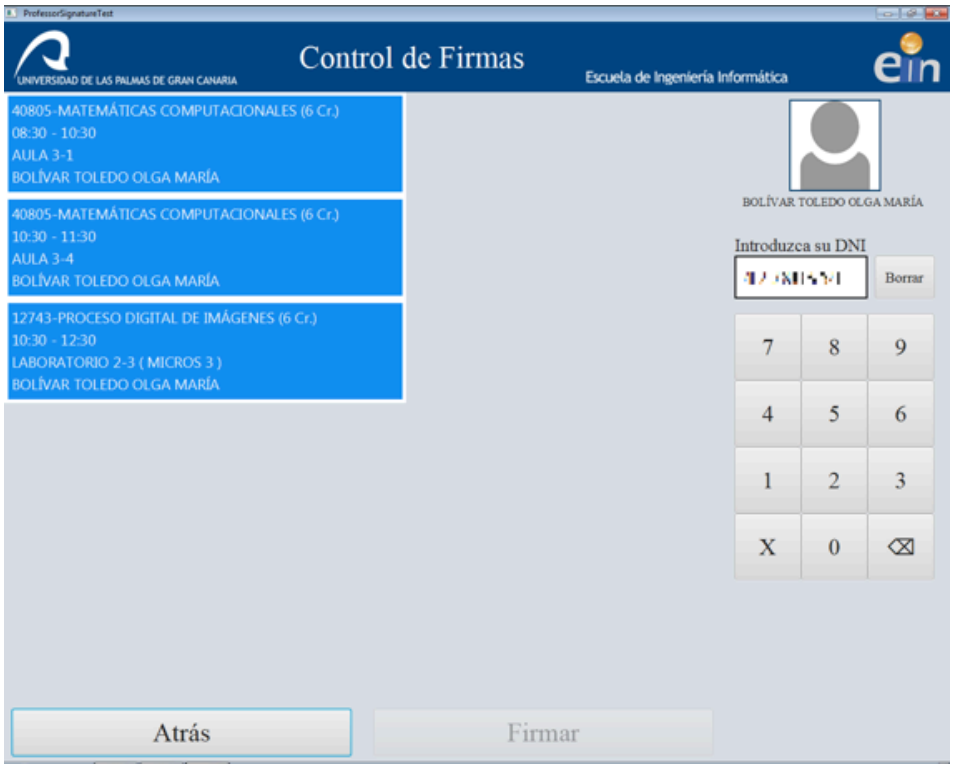
Al volver a la página de inicio, si pulsamos en “Sustitución” nos aparece la docencia actual, que para nosotros significa que la clase haya empezado como mucho hace 2 horas o dentro de 2 horas. Tenemos un espacio vacío a la derecha que se dejó en blanco para apoyar el dedo en esa zona y realizar un desplazamiento táctil.



Cuando seleccionamos asignaturas, nos permite la opción de firmar, la cual abre una cuadro de diálogo nuevo y activa la tableta digitalizadora.



Por último, si hubiese iniciado sesión con un profesor que tuviera docencia en ese momento, la pantalla de “Docencia del día” hubiese tenido este aspecto.

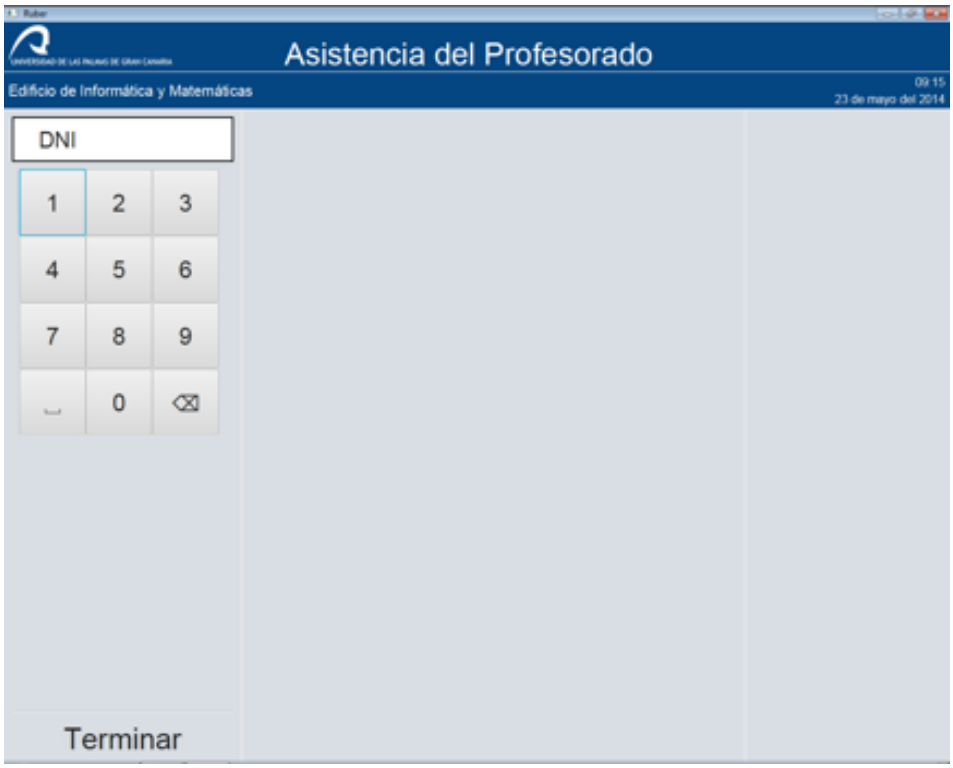


4.3.3. Iteración 2. Interfaz de usuario

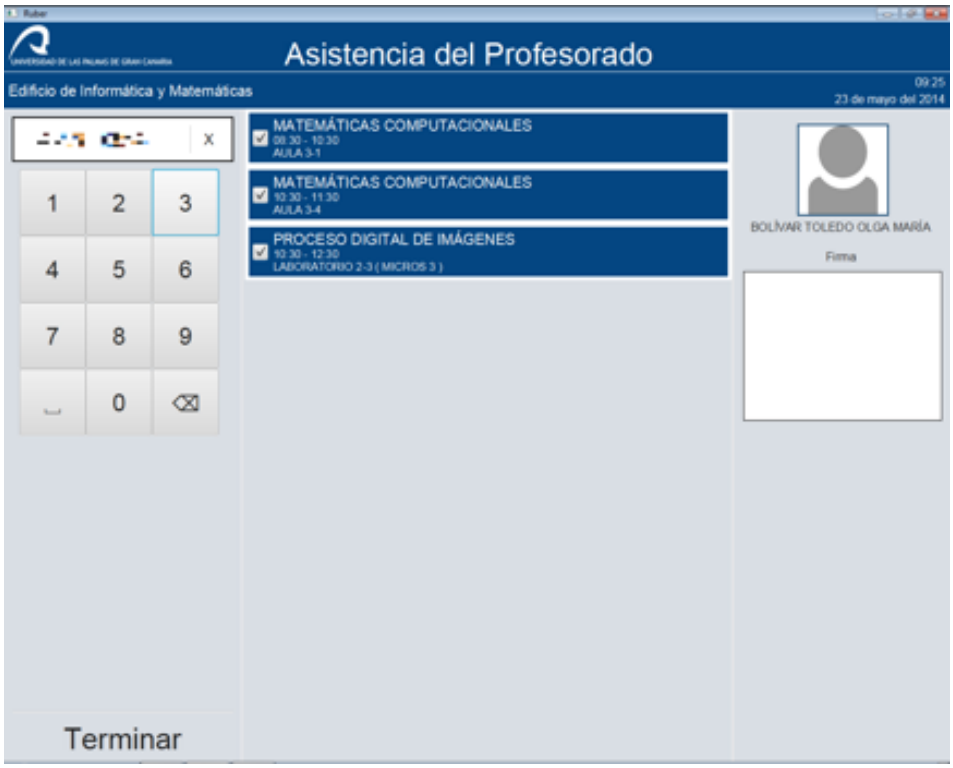
Una vez creado el primer prototipo con todas las funcionalidades que quería el *product owner*, nos reunimos con el tutor para conocer su opinión y recibir feedback ya que, a fin de cuentas, el tutor también es un profesor que acabará usando este sistema.

El tutor propone varios cambios en la interfaz a modo de hacerla más cómoda. Para empezar sugiere colocar el teclado a la izquierda, ya que nuestra forma de lectura es de izquierda a derecha, y colocar los números en orden normal como en una calculadora. También retiramos el concepto de "Control de firmas", ya que la palabra "control" puede tener una connotación negativa. Añadimos también la hora y el día como información adicional junto con el edificio en el que se encuentra el dispositivo. En esta primera versión mostramos siempre "Edificio de Informática y Matemáticas", pero cuando se implante en más escuelas universitarias se creará un diálogo que se muestre al iniciar el programa y permita elegir la escuela.

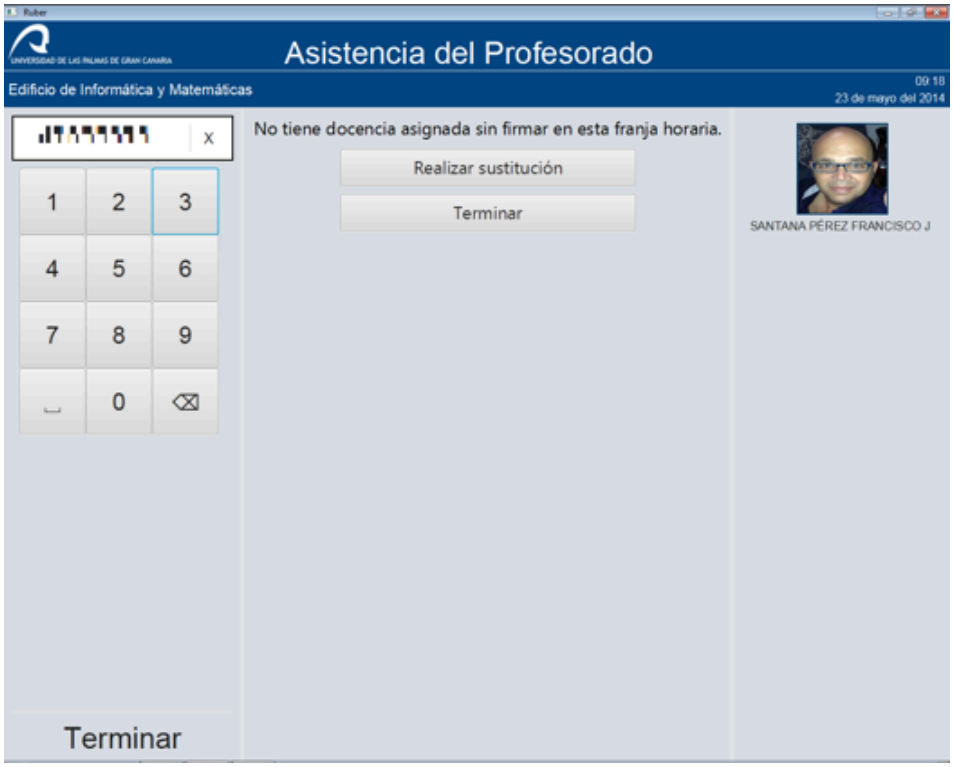
La pantalla principal queda ahora de esta forma:

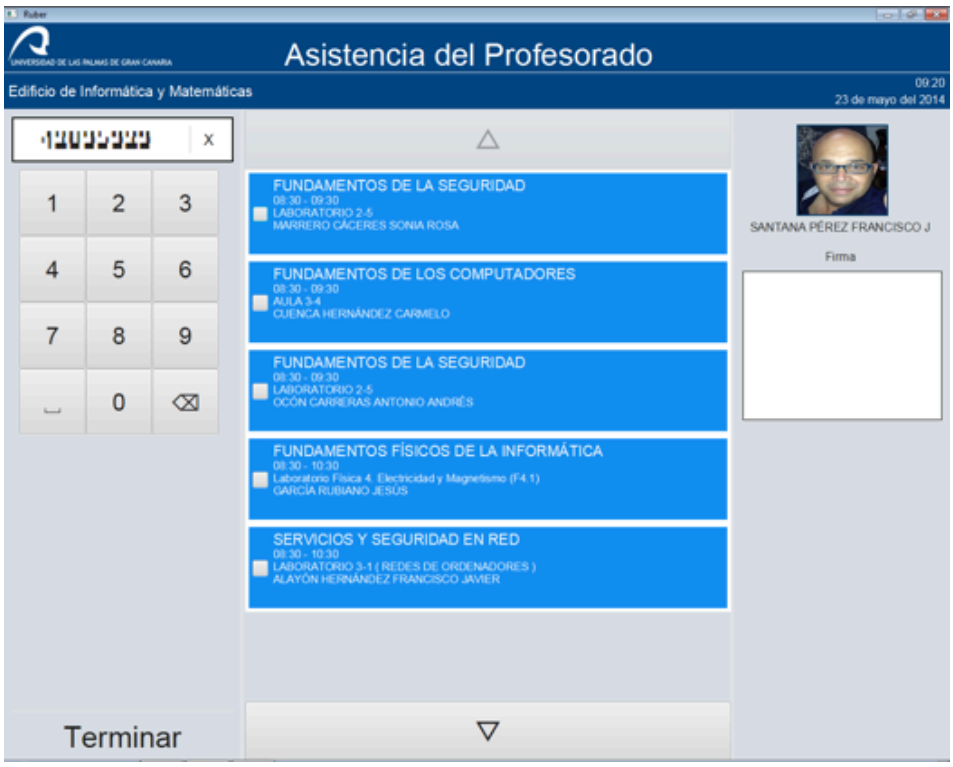


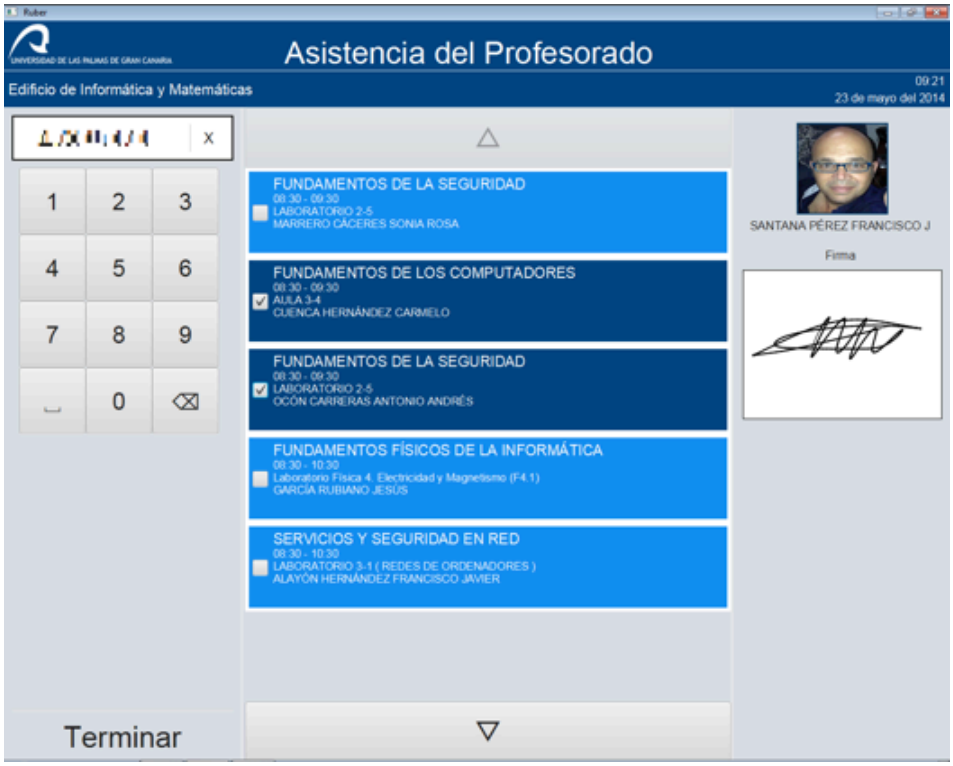
Una vez se inicia sesión, parece lógico pensar que no deberíamos dar la opción de firmar la docencia del día o sustituir. Si el profesor que inicia sesión tiene docencia hoy, primero debe firmar esa docencia. Una vez firmada (o si no tuviese), daremos la opción de sustituir a otro profesor. De esta forma, si iniciamos sesión y tenemos clase nos aparece lo siguiente:



En cambio, si ya hemos firmado o no tenemos docencia, nos dará la opción de sustituir.




Asistencia del Profesorado
Edificio de Informática y Matemáticas
09:20
23 de mayo del 2014
x
1 2 3
4 5 6
7 8 9
0
FUNDAMENTOS DE LA SEGURIDAD
08:30 - 09:30
LABORATORIO 2-5
MARRERO CÁCERES SONIA ROSA
FUNDAMENTOS DE LOS COMPUTADORES
08:30 - 09:30
AULA 3-4
CUENCA HERNÁNDEZ CARMELO
FUNDAMENTOS DE LA SEGURIDAD
08:30 - 09:30
LABORATORIO 2-5
OCÓN CARRERAS ANTONIO ANDRÉS
FUNDAMENTOS FÍSICOS DE LA INFORMÁTICA
08:30 - 10:30
Laboratorio Física 4. Electricidad y Magnetismo (F4.1)
GARCÍA RUBIANO JESÚS
SERVICIOS Y SEGURIDAD EN RED
08:30 - 10:30
LABORATORIO 3-1 (REDES DE ORDENADORES)
ALAYÓN HERNÁNDEZ FRANCISCO JAVIER
SANTANA PÉREZ FRANCISCO J
Firma
Terminar


Asistencia del Profesorado
Edificio de Informática y Matemáticas
09:21
23 de mayo del 2014
x
1 2 3
4 5 6
7 8 9
0
FUNDAMENTOS DE LA SEGURIDAD
08:30 - 09:30
LABORATORIO 2-5
MARRERO CÁCERES SONIA ROSA
FUNDAMENTOS DE LOS COMPUTADORES
08:30 - 09:30
AULA 3-4
CUENCA HERNÁNDEZ CARMELO
FUNDAMENTOS DE LA SEGURIDAD
08:30 - 09:30
LABORATORIO 2-5
OCÓN CARRERAS ANTONIO ANDRÉS
FUNDAMENTOS FÍSICOS DE LA INFORMÁTICA
08:30 - 10:30
Laboratorio Física 4. Electricidad y Magnetismo (F4.1)
GARCÍA RUBIANO JESÚS
SERVICIOS Y SEGURIDAD EN RED
08:30 - 10:30
LABORATORIO 3-1 (REDES DE ORDENADORES)
ALAYÓN HERNÁNDEZ FRANCISCO JAVIER
SANTANA PÉREZ FRANCISCO J
Firma
Terminar

Con todos estos cambios seguimos teniendo las mismas funcionalidades que habíamos hecho en la interfaz que vio primero el *product owner*, pero ahora lo tenemos todo de una manera más rápida.

Por poner un ejemplo, si antes un profesor quería firmar su docencia, tenía que introducir su DNI (8 clicks), pulsar el botón "Docencia" (1 click), seleccionar sus clases (2 por ejemplo), pulsar el botón "Firmar" (1 click), firmar y pulsar OK en la tableta (1 click) y pulsar "Terminar" en la pantalla (1 click). En total, el profesor tendría que pulsar entre pantalla y tableta unas 14 veces cada vez que vaya a firmar. Con esta nueva interfaz únicamente ponemos el DNI (8 clicks), nos muestra nuestras clases seleccionadas por defecto, firmamos y pulsamos OK en la tableta (1 click) y pulsamos "Terminar en pantalla" (1 click). Esto nos deja en 10 clicks, que además podemos reducir de la siguiente forma. Un DNI tiene 8 dígitos pero muchas veces con introducir 3 o 4 dígitos, ya nuestro número es único frente al resto de profesores, por lo que podríamos comprobar, cada vez que se introduce un dígito, si ya tenemos lo suficiente del DNI para identificar a un usuario. Con este método, podríamos dejar la media de clicks en 5 para firmar una docencia.

Con esto, dimos por concluida la iteración y pusimos a funcionar el producto ante otros profesores de la escuela.

4.3.4.Iteración 3. Feedback y mejoras

En general, los profesores estuvieron satisfechos con el sistema, salvo algún caso excepcional de profesores que no quisieron firmar por rechazo a firmar en tabletas digitalizadoras. Intuimos que para estos casos especiales habrá que proponer un método alternativo, ya sea firmar en papel o autentificar al usuario con el CAS como estaba pensado en un inicio.

Recibimos bastantes sugerencias del profesorado. Algunos cambios eran muy sencillos, como poner algún texto o botón más grande, o mostrar una notificación de que se ha firmado al acabar.

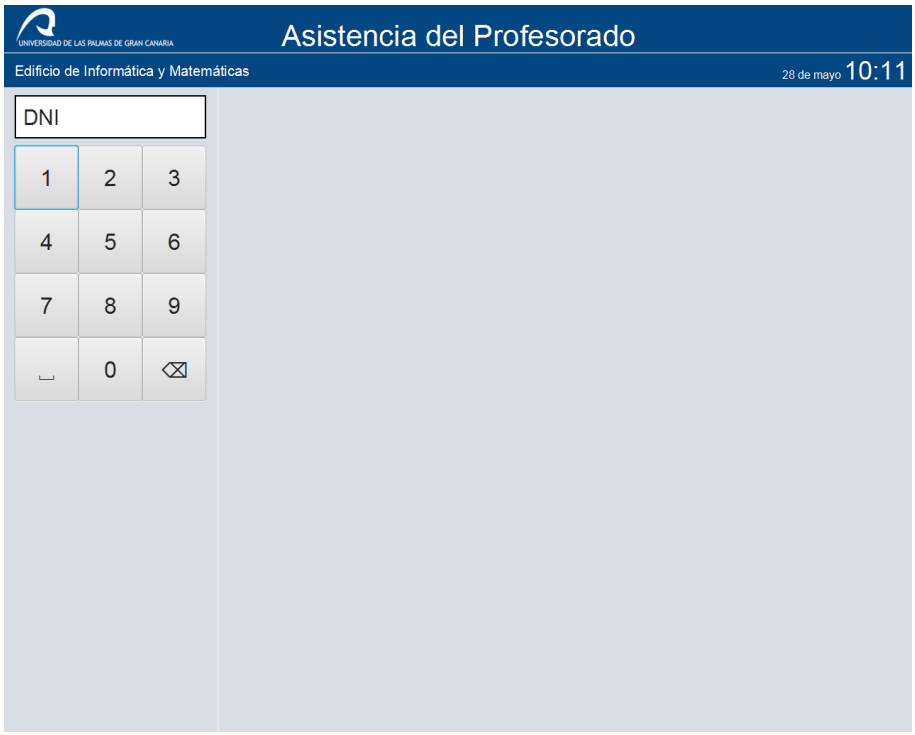
También hubo sugerencias relativas a la seguridad. Cuando se pensó inicialmente, la escritura predictiva del DNI parecía muy cómoda, ya que escribiendo 2 o 3 dígitos de tu DNI ya podías tenerlo completo. Sin embargo, esto acaba provocando problemas. Para empezar, puede ocurrir que sin querer nos equivoquemos al introducir el DNI y por un número mal nos complete el DNI de otro profesor, haciendo que tengamos que borrar todo y volver a empezar. Pero es que además, esto puede provocar que haya gente que se ponga a rellenar números al azar hasta que aparece un profesor y firma en su nombre sin que el profesor se haya enterado.

Por estos motivos, hemos decidido retirar esta funcionalidad de la siguiente versión. Además, para una versión futura se podría implementar la sugerencia de un profesor, que consiste en que haya que introducir el DNI y un símbolo o algún número más que solo conozca el profesor, a modo de contraseña muy simple para que no se pueda iniciar sesión en nombre de otra persona de manera tan fácil.

Otros profesores comentaron la idea de que el sistema les enviara un correo electrónico cuando se ha firmado la docencia. Esto es algo importante para el profesor, principalmente para dos casos. Si un profesor tiene que ir a un congreso y un compañero le va a sustituir, es interesante que el profesor que no va a estar presente recibiese un correo confirmándole que le han sustituido.

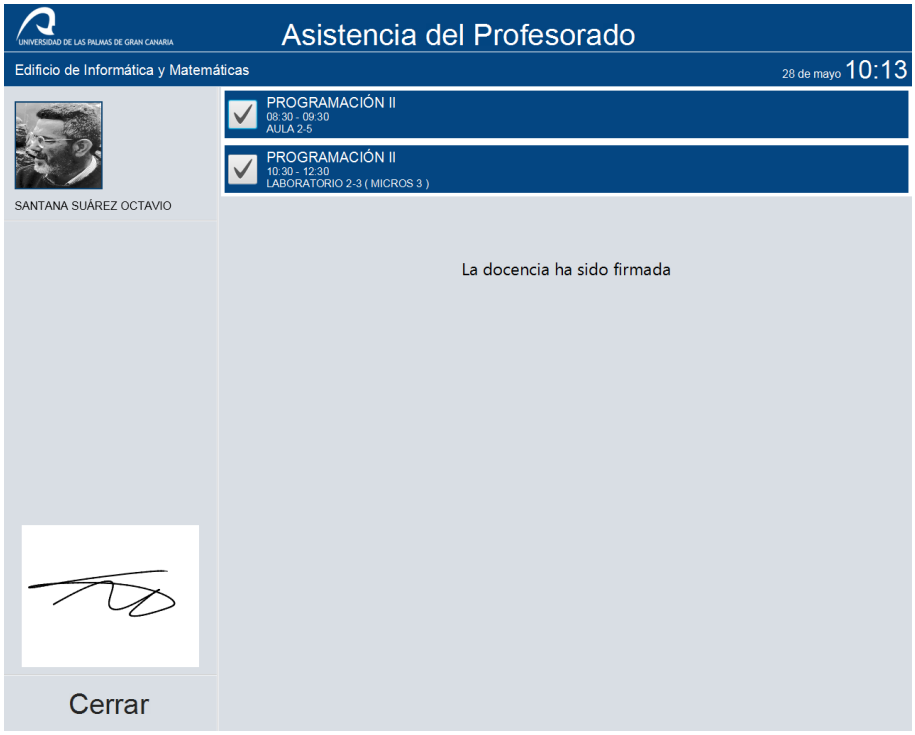
Nos reunimos con el tutor para poner en común todo el feedback y empezar a implementar los cambios pedidos. Todos estos cambios se pudieron implementar en muy poco tiempo, si bien habría que hacer una pequeña puntualización en el caso de la notificación por correo. La funcionalidad se encuentra implementada, enviando correos por defecto a las dirección institucionales de los profesores. Puede darse el caso de profesores que no deseen recibir estos correos o que deseen recibirlos en una dirección de correo alternativa, por lo que entendemos que habría que dar algún tipo de aplicación desde la que se puedan configurar dichos parámetros. A continuación, comentamos los cambios realizados en esta iteración.

Para empezar, llegamos a la conclusión de que no tiene sentido, una vez iniciada la sesión, mostrar el teclado para introducir el DNI y el nombre y foto del profesor, ya que el teclado ya no se va a usar. Por ello, hemos hecho lo siguiente:

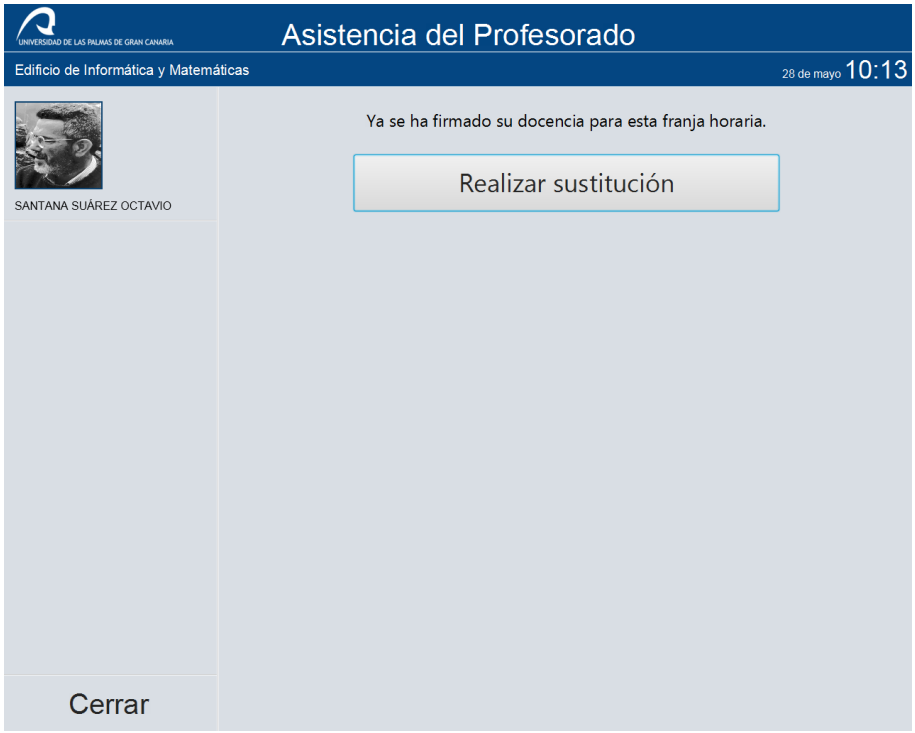


Una vez hemos iniciado sesión, el teclado y la pantalla de DNI desaparecen y solo se muestra la imagen del profesor junto con su nombre y un botón para cerrar la sesión.

Si el profesor tiene docencia, vemos todas sus asignaturas marcadas por defecto como ya hacíamos antes. Y una vez firmadas, se muestra un mensaje durante 3 segundos a modo de notificación:

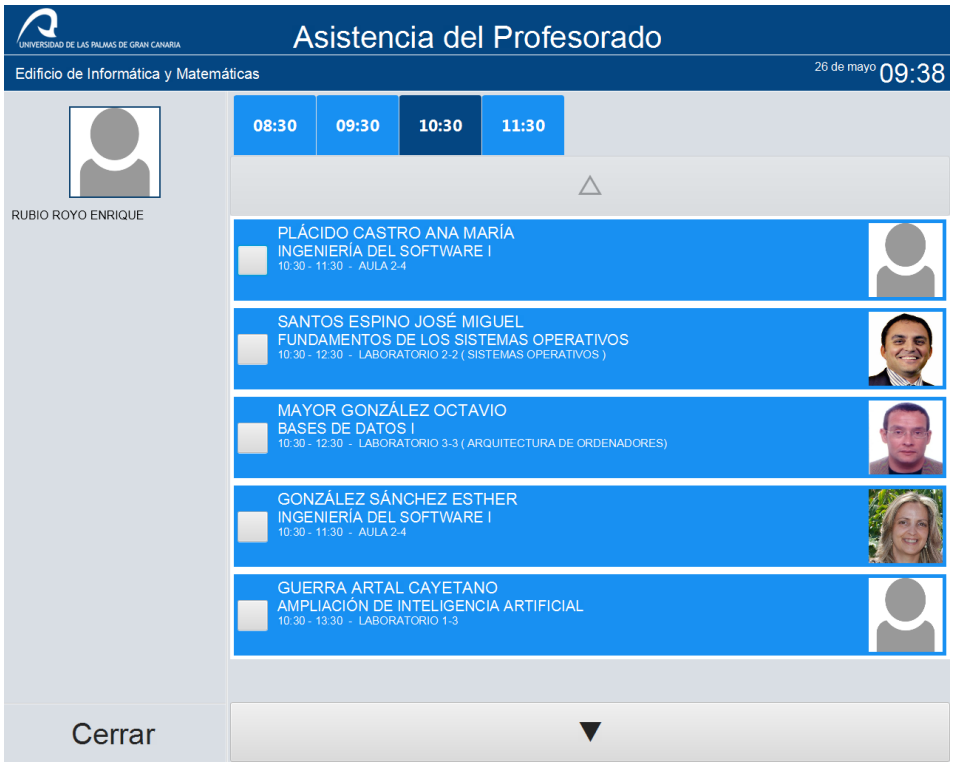


Si cerrásemos sesión y volviéramos a entrar con el mismo profesor, veríamos que ya se ha firmado nuestra docencia para esta franja horaria.

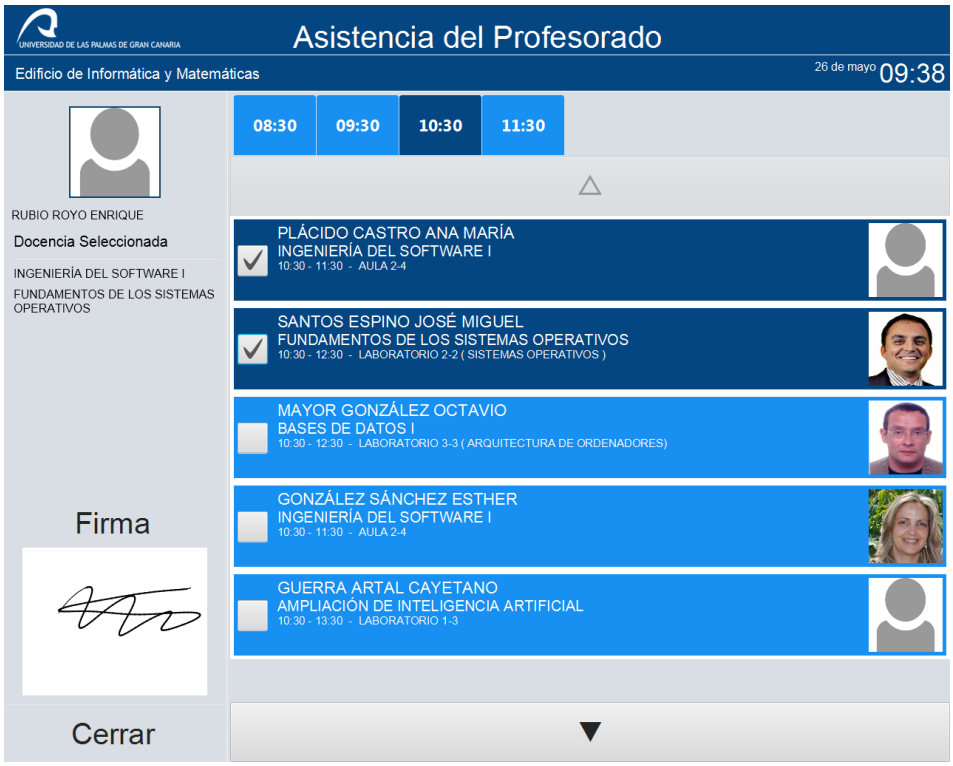


El caso de la sustitución si sufrió más cambios en esta iteración.

En un principio, nos planteamos agrupar las asignaturas por franjas horarias, de tal forma que la búsqueda es más rápida para el profesor que sustituye:



Además, al seleccionarse las asignaturas, se van mostrando en el panel de la izquierda a modo de recordatorio por si hemos ido cambiando de pestaña:

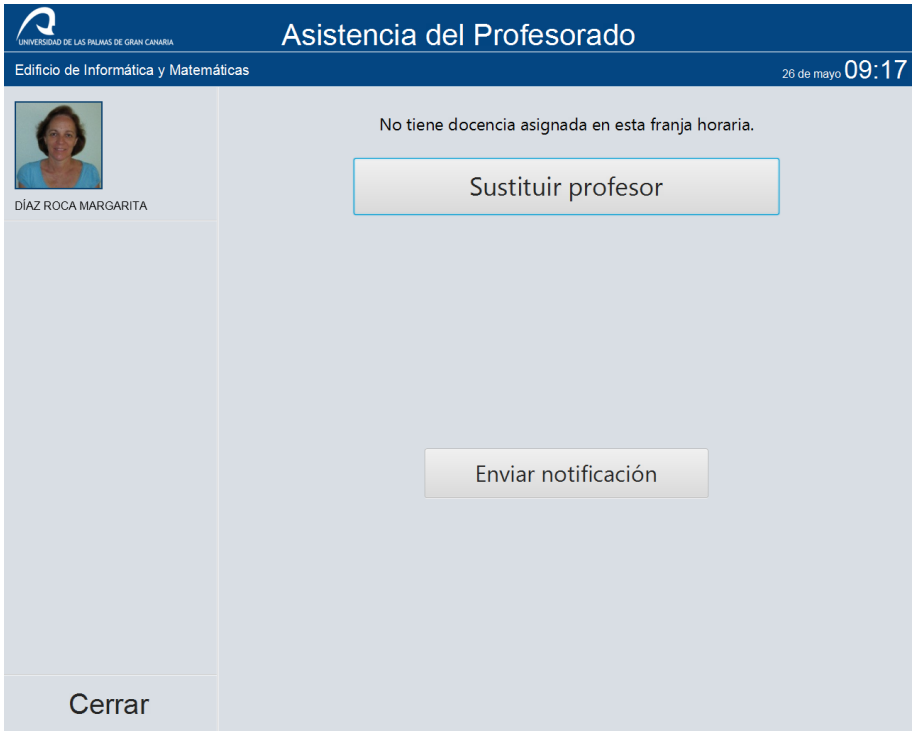


Después de hacer esto nos replanteamos por completo el proceso. En realidad, un profesor busca a otro profesor para sustituirle, no busca una asignatura para cubrir la docencia. Por lo que en realidad, parece lógico que lo que se debería mostrar sea una lista de profesores con docencia en la franja horaria actual. La interfaz quedaría de la siguiente forma:

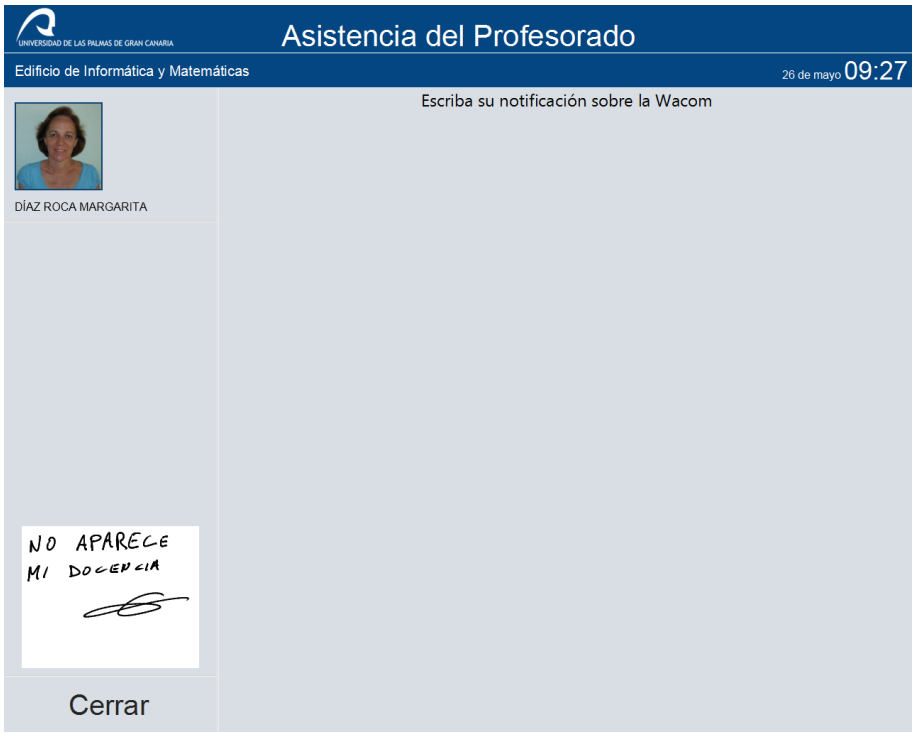
Una vez se ha seleccionado al profesor, se muestra su nombre y su foto debajo del profesor que ha iniciado sesión, y se lista la docencia que tiene el profesor sin firmar:

En una de las reuniones finales con el *product owner*, uno de los profesores propuso que la aplicación permitiera enviar notificaciones, en el sentido de que a principio de curso se da mucho que hay problemas con los horarios de docencia y los profesores no tienen su docencia en la hoja. Al *product owner* le gustó la idea y vimos que incluso sobre la misma tableta digitalizadora se podía escribir un mensaje corto sin mucha complicación.

De forma rápida implementamos esta funcionalidad, que quedó así:



Al iniciar sesión nos aparece un botón de "Enviar notificación", al pulsar en él nos pide que escribamos nuestro mensaje sobre la tableta y al pulsar terminar, la notificación quedará registrada.



5. Conclusiones

La realización del trabajo ha sido bastante satisfactoria ya que se han podido cumplir todos los objetivos propuestos en un principio.

Desde el punto de vista general del desarrollo de un producto, ha sido interesante ver como ha surgido la propuesta de valor para los usuarios a partir de ellos y no de los desarrolladores. En un principio partíamos de un producto que a los usuarios no aportaba nada; aportaba valor al *product owner* por los datos que recopilaba, pero no a las personas que usaban el producto. Sin embargo, la puesta en marcha de un primer prototipo y la recolección de feedback por parte de los usuarios hicieron posible descubrir una propuesta de valor que motiva a los usuarios a usar el producto.

Pasando al desarrollo del producto software, ha sido enriquecedor el desarrollo de un producto real, en el que hemos tenido que ir cambiando las características que habíamos pensado en un principio para adaptarnos a los distintos problemas que aparecían. Además, se ha podido utilizar en parte la metodología de desarrollo guiado por pruebas (TDD) en el desarrollo del modelo de la aplicación.

Desde un punto de vista de aportaciones administrativas que da el software, se han conseguido implementar las funcionalidades principales que pretendíamos, dando la posibilidad de registrar la asistencia de un profesor a una clase o la sustitución de un profesor a otro.

El método de desarrollo elegido, basado en el agilismo, ha sido muy provechoso ya que es una manera muy cómoda de trabajar. No definimos en un inicio un conjunto rígido de requisitos y pasados 6 meses entregamos un producto final al *product owner*. En lugar de eso, el *product owner* iba viendo semanalmente lo que se iba haciendo y validando las decisiones que se iban tomando, lo que hacía el cambio más rápido que hacer muchos cambios cuando ya el producto está "terminado".

Entrando en concreto con el desarrollo guiado por pruebas, el hecho de tener pruebas automáticas nos ha dado la seguridad para realizar cambios en el sistema sin miedo, puesto que teníamos los tests que nos detectarían si ha habido un fallo. A modo de resumen, y por relacionar un poco con la descripción que se dio en apartados anteriores, yo diría que hacer TDD me ha aportado:

- Poder comprobar que todo funciona desde el principio, sin necesidad de tener un método *main* o una interfaz de usuario.
- Poder refactorizar sin miedo algunas secciones de código que en principio eran menos "limpias".
- Saber por donde empezar a trabajar y tener un ritmo de trabajo más o menos constante.

Me hubiese gustado poder aplicarlo al desarrollo de la interfaz, ya que había momentos en los que sí note que hacía cambios en la interfaz con más "miedo" porque tenía que hacer las pruebas a mano y acordarme de todo lo que tenía que probar cada vez que hacía un cambio ya que no tenía pruebas automáticas para validar. En futuros trabajos sería interesante aplicar esta metodología al desarrollo de los otros módulos de una aplicación (interfaz de usuario, control).

Todo el diseño de la interfaz de usuario se debatió bastante con ambos tutores, y se puede ver como la interfaz tuvo muchas versiones distintas. A lo largo de todas estas versiones hemos intentado conseguir que el software sea lo más usable posible, sin necesidad de un manual de usuario. De hecho, cuando se puso en funcionamiento a modo de prueba, los profesores no tuvieron demasiada dificultad para utilizarlo, y las pocas dificultades que encontraron nos las comentaron para que las pudiéramos solucionar.

A nivel del código desarrollado, se ha intentado conseguir una arquitectura limpia, con clases que muy rara vez superan las 100 líneas de extensión. Igualmente, no es nuestra intención llegar a una arquitectura y no volver a modificarla nunca. La arquitectura aún puede mejorarse con la intención de hacerla más fácil de entender para futuros desarrolladores que mantengan el programa. Esas es una de las aportaciones que nos da el TDD, poder cambiar el programa sabiendo que no hemos roto nada.

6. Trabajos futuros

El desarrollo de este sistema no acaba en el ámbito de este Trabajo de Fin de Grado. Aún faltarían por desarrollar una serie de módulos que hagan uso de la aplicación que se ha desarrollado.

Para empezar, habría que crear una aplicación que, con los datos recogidos a lo largo de un día lectivo, genere un fichero PDF con las firmas de todos los profesores que han impartido docencia en el día actual.

También podría ser interesante tener otro pequeño módulo que lo único que haga sea comprobar al final del día si ha habido alguna falta y notificar automáticamente al responsable. Esto también se podría hacer manualmente haciendo una consulta a la base de datos, pero esta forma podría estar automatizada por un lado la creación de PDF's para registrar la asistencia y el responsable del centro solo tiene que preocuparse cuando reciba un correo notificándosele de que ha habido alguna ausencia.

Por último, de cara a integrar el sistema con toda la universidad, lo ideal sería que el Servicio de Informática de la ULPGC pudiera darnos acceso directo a las bases de datos de la universidad, ya que tendríamos la información actualizada en todo momento. Con el sistema actual, si hubiese un cambio en los horarios, el director de la escuela nos lo tendría que notificar para nosotros actualizar la base de datos interna a la aplicación. Si ya estamos conectados directamente a la universidad, la actualización es inmediata.

Con respecto al trabajo realizado, también se podrían definir una serie de modificaciones futuras.

Por ejemplo, el trabajo se ha realizado sobre un ordenador con sistema operativo Windows 7. Uno de los posibles trabajos sería probar la aplicación en otros sistemas operativos para evitar posibles problemas que puedan surgir si al implantarse el sistema en otra escuela universitaria no se usara el mismo sistema operativo.

Por este mismo motivo también se podrían realizar pruebas con otras configuraciones software, como por ejemplo, modelos distintos de tabletas digitalizadoras, para buscar algunas soluciones hardware más baratas o que los controladores estén disponibles en más sistemas operativos.

También podrían añadirse otras funcionalidades, como por ejemplo a la hora de sustituir a otro profesor, dar más opciones de filtrado, como el área de conocimiento o un histórico de profesores para que estén más cerca los profesores que son más probables para sustituir.

Con respecto a las notificaciones, también se podrían hacer modificaciones para elegir el tipo de notificación y hacer un filtrado más efectivo para el responsable de revisar esos datos. De primeras se nos podría ocurrir notificaciones de "Fallo en la docencia" o "Problema en el aula".

Por último, habría que dedicar un tiempo a formar personal para administrar la aplicación si pasase a mantenerla el Servicio de Informática de la ULPGC. Se ha intentado que la arquitectura sea lo más limpia posible para que si el código lo coge otro desarrollador no tenga demasiados problemas, pero siempre es mejor si ese nuevo desarrollador tiene a mano al autor original para comentar dudas. Además, considero que es un proceso beneficioso también para el desarrollador original, ya que recibirá opiniones de otro programador y podrá aprender cosas nuevas para hacer su código más legible.

7. Fuentes de Información

A continuación, se citan las fuentes de información utilizadas para el desarrollo del proyecto:

Relativas al uso de JavaFX

- Layout
  docs.oracle.com/javafx/2/layout/builtin_layouts.htm
- CSS
  docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html
- Eventos
  docs.oracle.com/javafx/2/api/javafx/event/Event.html
- Imágenes
  stackoverflow.com/questions/16990482/java-lang-illegalargumen
- Cambios en la IU desde un hilo distinto al principal
  stackoverflow.com/questions/13784333/platform-runlater-and-task-javafx
- Controladores desde FXML
  stackoverflow.com/questions/12543487/javafx-nested-controllers-fxml-include
- Eliminar barra superior de la ventana de la aplicación
  stackoverflow.com/questions/9861178/javafx-primarystage-remove-windows-borders
- Texto en varias líneas cuando no cabe en una sola
  stackoverflow.com/questions/20743668/word-wrapping-inside-a-titledpane-that-is-inside-vbox
- Capturar cualquier evento en la aplicación
  docs.oracle.com/javafx/2/events/processing.htm
- Capturar evento al cerrar aplicación
  www.java2s.com/Code/Java/JavaFX/Stagecloseevent.htm

Otras fuentes referentes al desarrollo

- Envío de correo desde Java
  www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/
- Cargar una DLL en Java
  blog.cedarsoft.com/2010/11/setting-java-library-path-programmatically/

Relativas a normativa y legislación

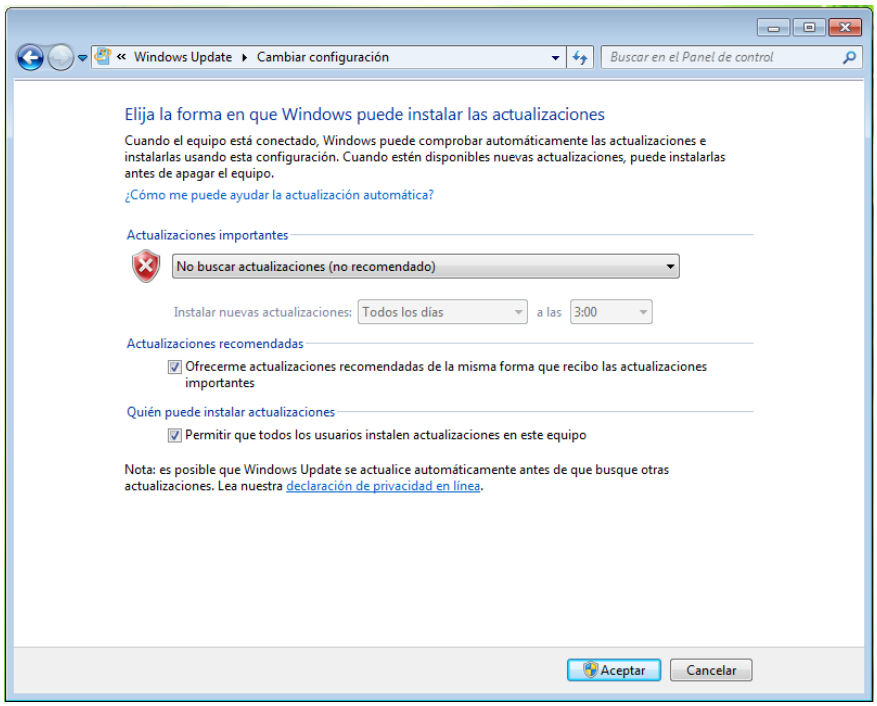
- Control de asistencia ULPGC
  www.ulpgc.es/hege/almacen/download/18/18456/normas_para_el_control_de_asistenci.pdf
- Ley Orgánica de Protección de Datos
  http://www.agpd.es/portalwebAGPD/jornadas/dia_proteccion_2011/responsable/index-ides-idphp.php

Anexo A. Manual de instalación

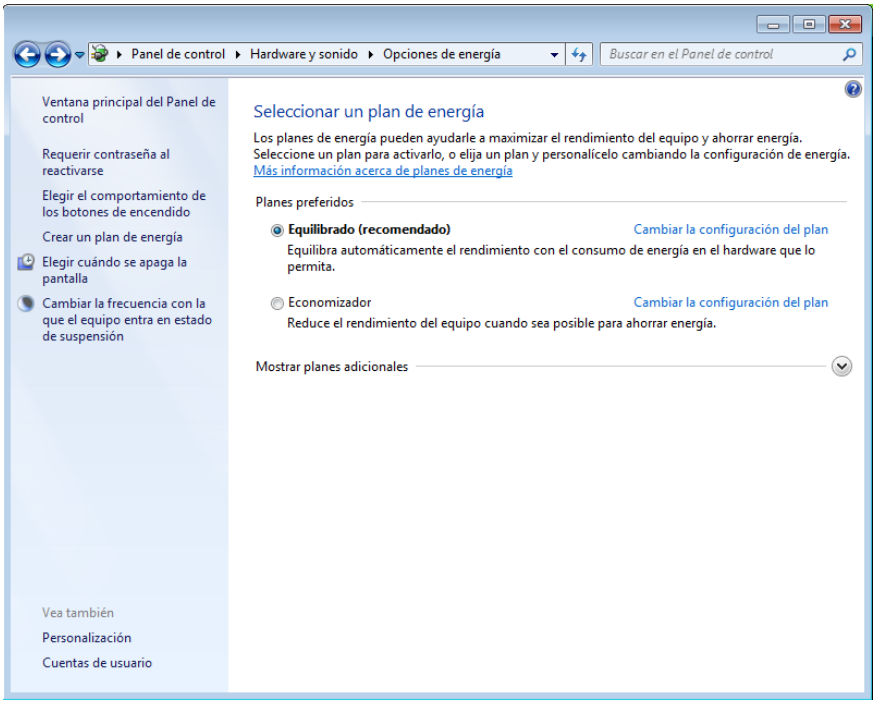
La instalación del software en una máquina es un proceso muy sencillo y que se describe a continuación.

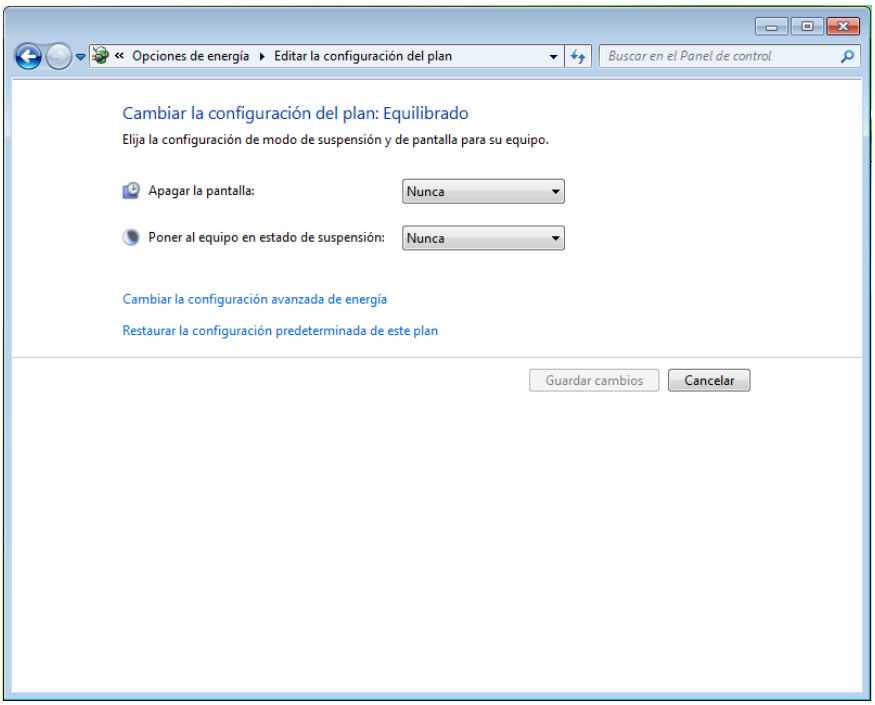
Lo primero que recomendamos es que se desactiven las actualizaciones automáticas en el equipo con Windows, para evitar reinicios inesperados. También sería deseable desactivar el apagado de pantalla automático para que se muestre siempre la pantalla inicial de la aplicación y no se quede en negro.

En Windows 7, para desactivar las actualizaciones automáticas, debemos ir al Panel de Control y entrar al menú de Windows Update, en la opción *Activar o desactivar la actualización automática*.



Para desactivar el apagado de la pantalla, también debemos ir al Panel de Control, en Opciones de energía. Desde ahí podemos cambiar la configuración del plan de energía y establecer que la pantalla no se apague nunca y que el equipo no entre en reposo.





Como requisitos antes de poner en funcionamiento la aplicación, el equipo debe de tener instalado el STU SDK de Wacom en el directorio "C:\Program Files". Tampoco debemos olvidarnos de tener conectada en todo momento la tableta Wacom para realizar las firmas.

A la hora de arrancar la aplicación, ésta se encuentra dentro de la carpeta "dist" la cual contiene las librerías necesarias y las imágenes de perfil del profesorado. Haciendo doble click sobre el fichero Ruber.jar, la aplicación debería arrancar sin ninguna complicación.