Universitat Oberta de Catalunya

CORE STUDIO Pilates

Treball fi de Grau: Sistema de gestió de CORESTUDIO Pilates Memòria final

Ignacio González Bullón
UOC 2015/16

Índex

Introducció

Prefaci

Aquest document és la memòria final del Treball Fi de Grau del Grau d'Enginyeria del Programari. L'objectiu principal plantejat a l'hora de definir el projecte és posar en consonància gran part de les competències adquirides durant els estudis de Grau per a adaptar-los a un projecte real fent ús de tecnologies molt presents i actuals en el mercat i que suposin un repte d'estudi i una oportunitat per a l'aprenentatge global. Atenent aquest objectiu s'ha aprofitat l'oportunitat sorgida de la necessitat d'una petita empresa en creixement per a elaborar aquest projecte com una aplicació web, que suposa una activitat prou transversal com per a implicar molts dels coneixements adquirits durant el Grau, ja que implica: treball al frontend on s'han d'aplicar competències relacionades amb la **interacció persona – ordinador**; treball al backend on es poden aplicar competències relacionades amb els **sistemes distribuïts**, la **seguretat a les xarxes**; treball amb **bases de dades**; i treball de **gestió del projecte** i tractament amb el **client**.

El projecte consisteix en una aplicació web empresarial feta amb tecnologies Java i Javascript principalment degut tant a la seva àmplia demanda en el mercat tanmateix com per la seva robustesa. Java, concretament, és el llenguatge que encapçala l'índex Tiobe i és un llenguatge molt sòlid pel nombre d'anys que porta al mercat, la seva gran comunitat i el elevadíssim nombre solucions i frameworks construïts al seu voltant. D'altra banda tenim Javascript, un llenguatge imprescindible en el costat client de les aplicacions web que, a més, està creixent gràcies als frameworks que faciliten la creació d'aplicacions web amb una part client més pesada i perquè comença a ser una alternativa del costat servidor amb la presència de Node.

La idea és desacoblar tant com sigui possible la lògica de negoci que es duu a terme als servidors de la part client, dedicada a interactuar amb l'usuari. És per això que s'ha decidit separar aquestes parts totalment reduint la part Java a un conjunt de serveis REST amb els que l'aplicació ha d'accedir per a fer les operacions necessàries, deixant la gestió de la part client i la interacció amb l'usuari com una responsabilitat exclusiva de l'aplicació web. Per tant, es prescindeix de molta part del stack JavaEE per a aconseguir aquest desacoblament (no es fa ús de JSF o JSP), d'aquesta manera l'aplicació és més flexible a adaptar-se a suports diferents com pot ser una aplicació mòbil.

Visió del projecte

Corestudio Pilates és una empresa dedicada a impartir classes de Pilates en totes les seves variants. L'empresa ha crescut en els darrers mesos en nombre de clients i treballadors, cosa que ha fet inviable continuar amb el procés manual de control d'assistència i de pagaments basat en un conjunt de fulles de càlcul. És per això que l'estudi requereix d'un nou sistema que permeti automatitzar tot aquest procés.

El producte a entregar deu, si més no, ser capaç de gestionar els clients i els seus pagaments, contemplant tots els tipus d'abonament que ofereix l'empresa i els descomptes que s'apliquen a determinats clients. Així mateix, s'ha d'observar la forma en que es gestionen els abonaments, tant els individuals com els de grup, per a portar a terme un control exhaustiu dels mateixos. El sistema ha de ser prou flexible com per a contemplar diferents situacions en el transcurs d'un abonament (absències justificades, recuperacions de classes perdudes, classes extra, festius, etc.) i ajustar les dates en conseqüència. Igualment, ha de disposar d'un mecanismes per a notificar a l'usuari (treballador de l'estudi) sobre abonaments que estiguin a punt d'exhaurir.

D'altra banda, el sistema ha d'integrar tota aquesta informació en un espai de comptabilitat mensual on es puguin obtenir totes les dades que necessita la gestoria per a fer la comptabilitat de l'empresa. Aquí s'han de contemplar, tant els ingressos derivats de les classes (ajustant les classes abonades al mes en que es consumeixen) com tot tipus de despesa. Aquesta secció, a més, ha de produir informació interanual per a estudiar l'evolució del negoci i les aportacions de cada tipus d'ingrés (grups, individuals, modalitats de pilates, etc.).

D'altra banda, es requereix que el producte funcioni sense connexió a Internet donat el fet que l'estudi no disposa de connexió a Internet. A més, ha de poder córrer sobre un portàtil de gama mitjana. Igualment, s'ha de considerar algun mecanisme de protecció contra la pèrdua de dades.

Anàlisi de requisits

Definició dels stakeholders

Tal com s'ha indicat anteriorment, aquesta aplicació va destinada a una empresa petita (quatre treballadors). És per això que el nombre de persones interessades en el projecte és limitat:

- **Propietari**: el propietari de Corestudio. És qui ha impulsat el projecte degut a que el sistema de gestió manual no era adient per al volum de negoci que està adquirint l'estudi. També fa de professor i el seu interès és tenir una eina senzilla que li permeti obtenir tota la informació necessària per a gestionar el seu negoci i que permeti a la resta de professors introduir dades com els pagaments dels alumnes per alleugerir la seva càrrega de treball.
- **Professor**: és la persona encarregada d'impartir les classes de Pilates. Actualment també fa els cobraments als alumnes registrant-los en un albarà. Vol una eina que no li suposi una càrrega addicional de treball perquè el temps entre classes, que és quan els alumnes paguen, és molt reduït.
- **Alumne**: és la persona que va a rebre les classes de Pilates. Vol que no hi hagi errors amb els cobraments i que pugui consultar amb els professors l'estat del seu abonament amb facilitat.

Atenent l'anterior definició de stakeholders, es decideix que, per a definir els requisits, ens basarem en la informació que provingui del propietari ampliada per l'opinió dels professors. L'opinió dels alumnes quedarà més de banda perquè la seva interacció amb el sistema serà mínima.

Definició inicial de requisits

A partir d'entrevistes amb el propietari de Corestudio, amb el coneixement del negoci i el tipus de problema a resoldre es fa la primera definició general de requisits, tant funcionals com no funcionals.

Requisits funcionals

- ***Registre al sistema***: amb diferents perfils (usuari o treballador normal i administrador).
- ***Gestió de clients***: a més de la creació, recuperació, actualització i esborrament d'usuaris, s'ha de poder assignar un usuari a un grup, s'ha de poder mostrar les classes pendents de consumir, gestionar els seus pagaments o l'assistència a classe.
- ***Gestió de pagaments***: gestionar els pagaments dels clients, tenint en compte la data de pagament per a poder calcular la duració de l'abonament i distribuir el pagament en els mesos corresponents, el tipus d'abonament al que es correspon el pagament, per tal d'obtenir dades sobre els beneficis que genera cada tipus. També cal tenir en compte els canvis en els abonaments o la possibilitat de congelar un abonament per determinades circumstàncies.
- ***Gestió dels abonaments***: per afegir o modificar els tipus abonaments.
- ***Gestió de professors***: que inclou la possibilitat de mantenir notes sobre el professor, assignar un professor a una classe o grup o calcular el seu sou.
- ***Gestió de despeses***: que inclou la generació d'informes per a la gestoria.
- ***Gestió de les classes***: incloure un tauler on penjar sessions, exercicis, explicacions, horaris i altra informació important.

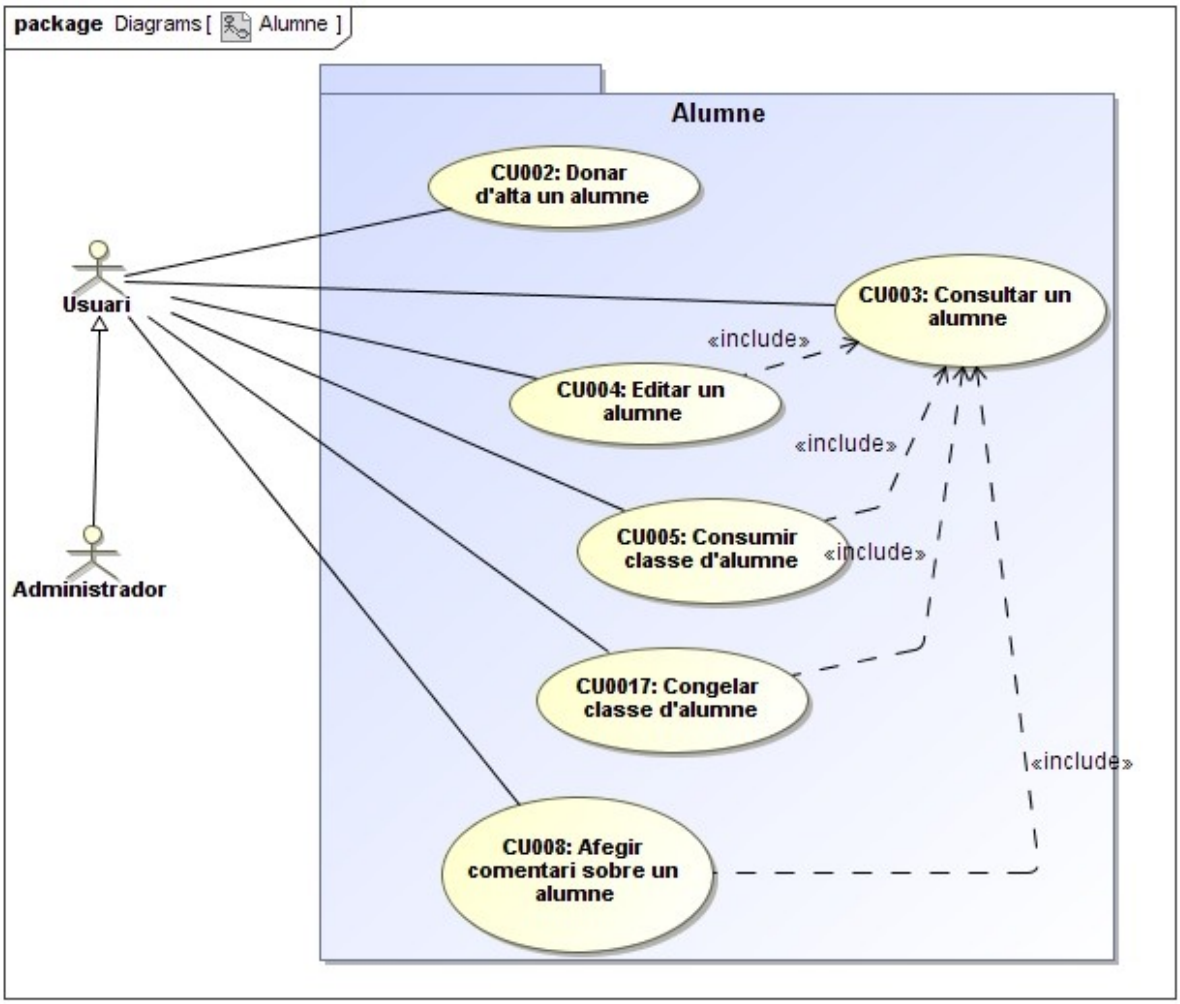
Requisits no funcionals

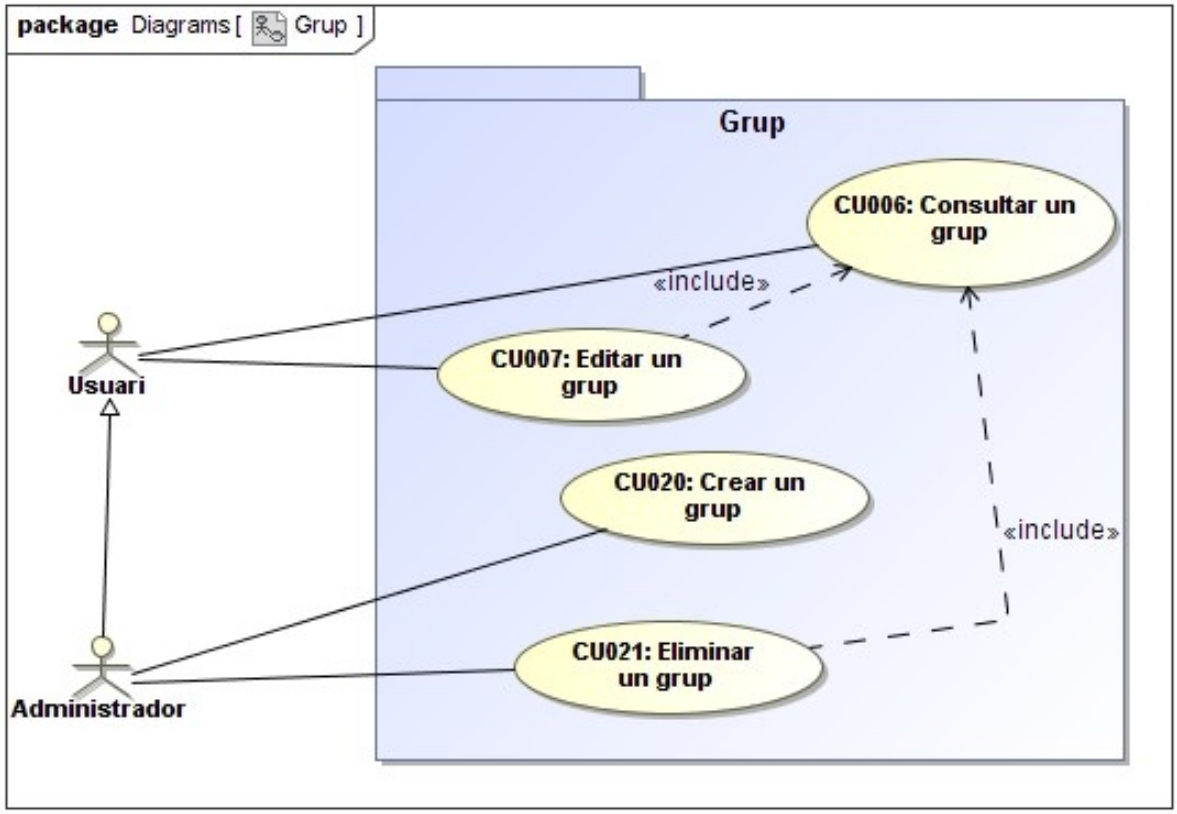
- L'aplicació ha de funcionar offline però ha de ser prou flexible com per a fer-la accessible des d'Internet (idealment des d'una tauleta).

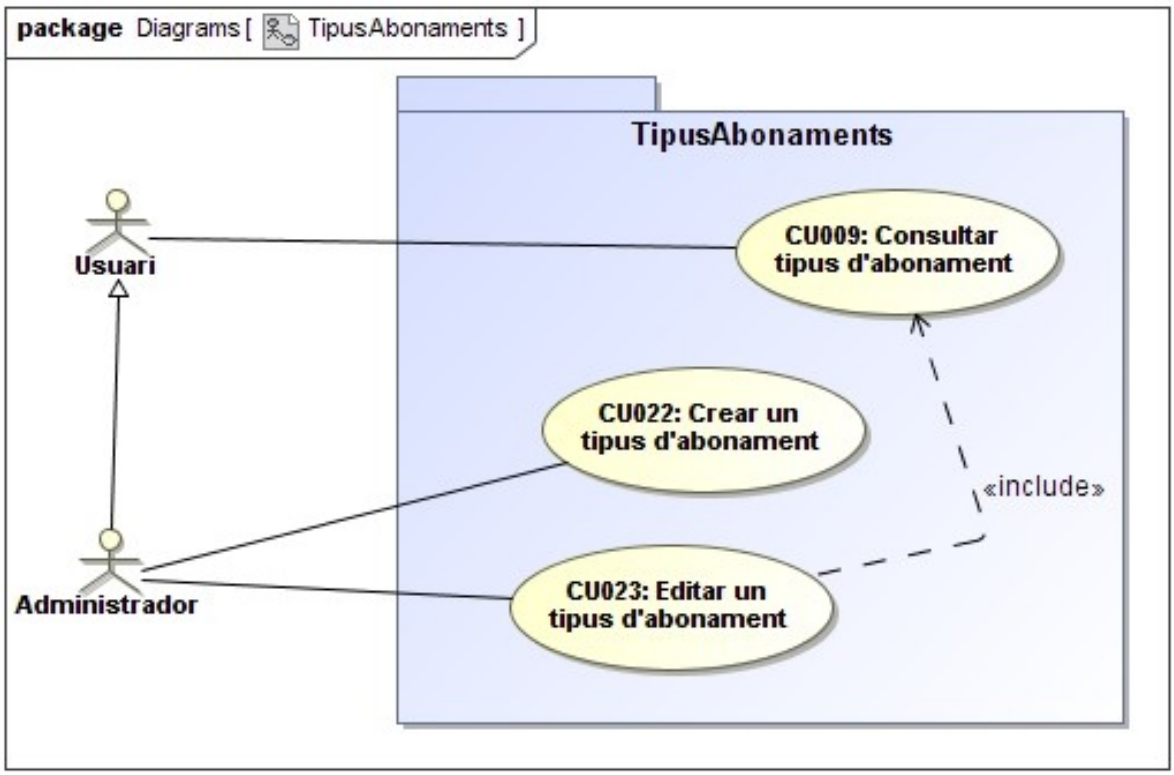
- L'aplicació ha de ser prou lleugera per a que es pugui desplegar en un portàtil de gama mitja.
- Ha de ser fàcil de fer servir i ha de mostrar la informació de manera prou entenedora com perquè es pugui mostrar als clients quan hi hagin dubtes.

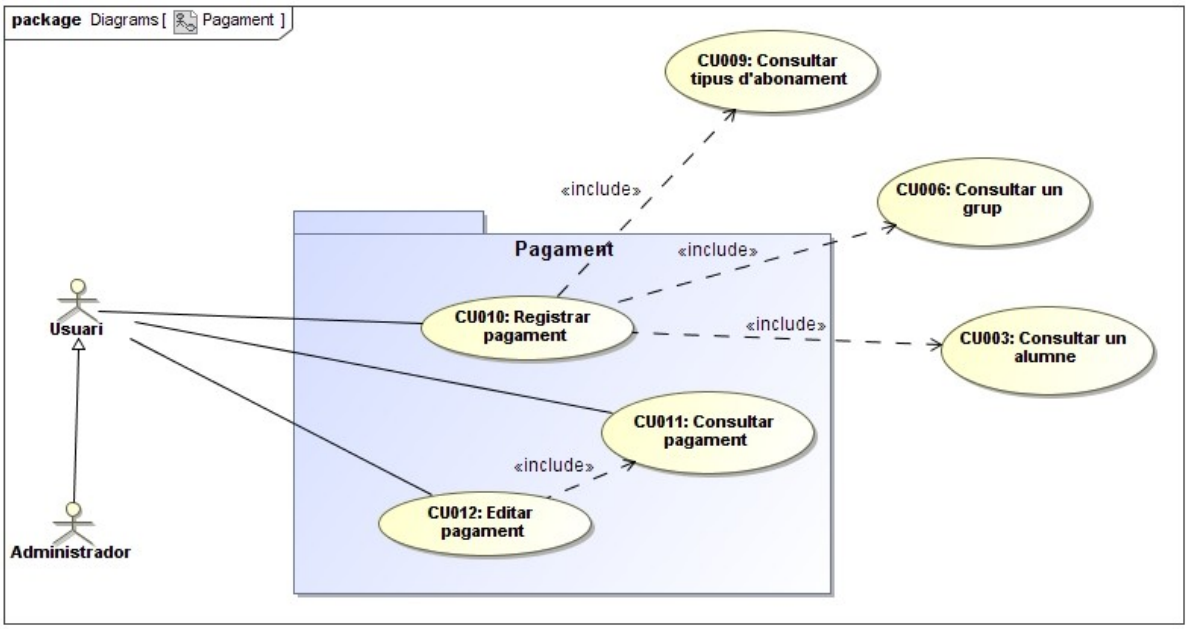
Diagrames de casos d'ús

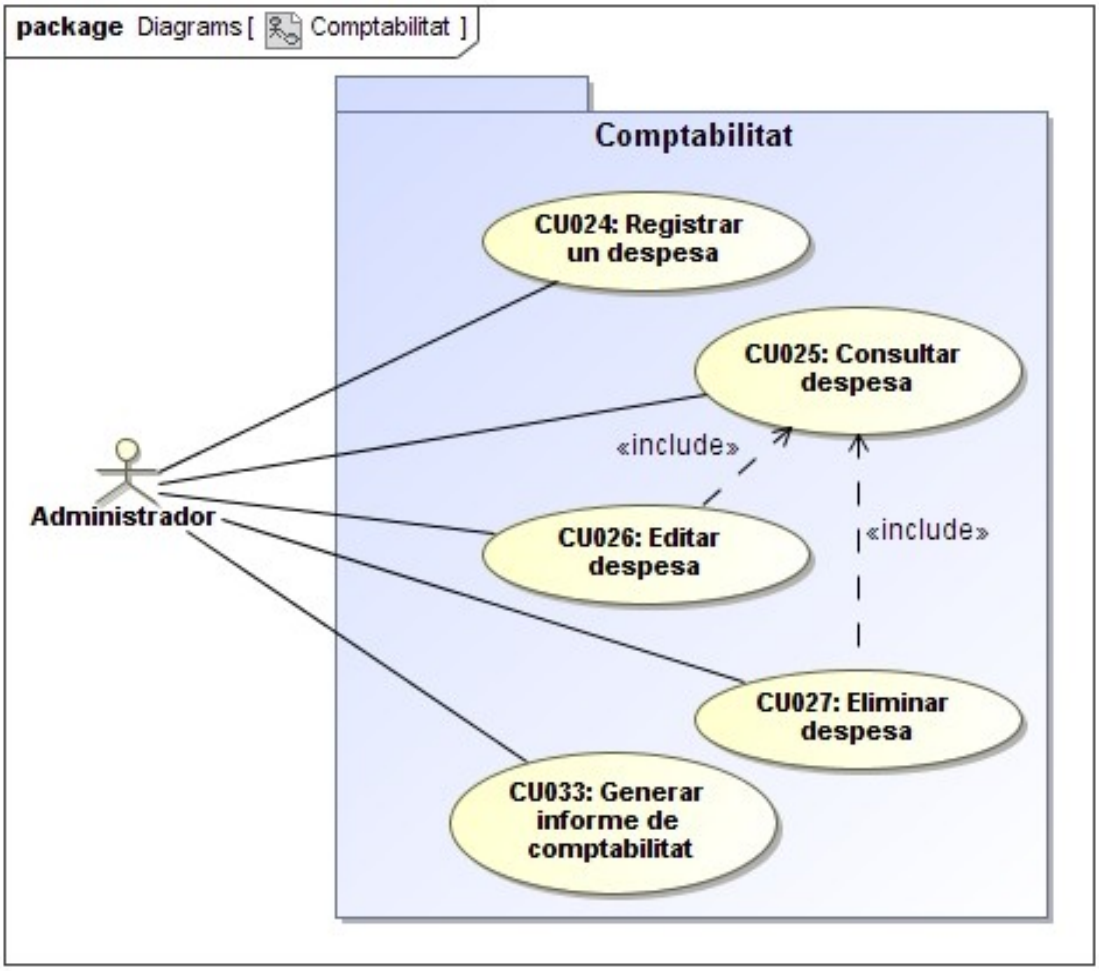
Partint d'aquesta primera informació pels casos d'ús, es fa una proposta general de casos d'ús a completar durant el temps del projecte. Atenent els requisits s'han definit dos actors, usuari i administrador, per a l'assignació dels següents casos d'ús dividits en grups funcionals.

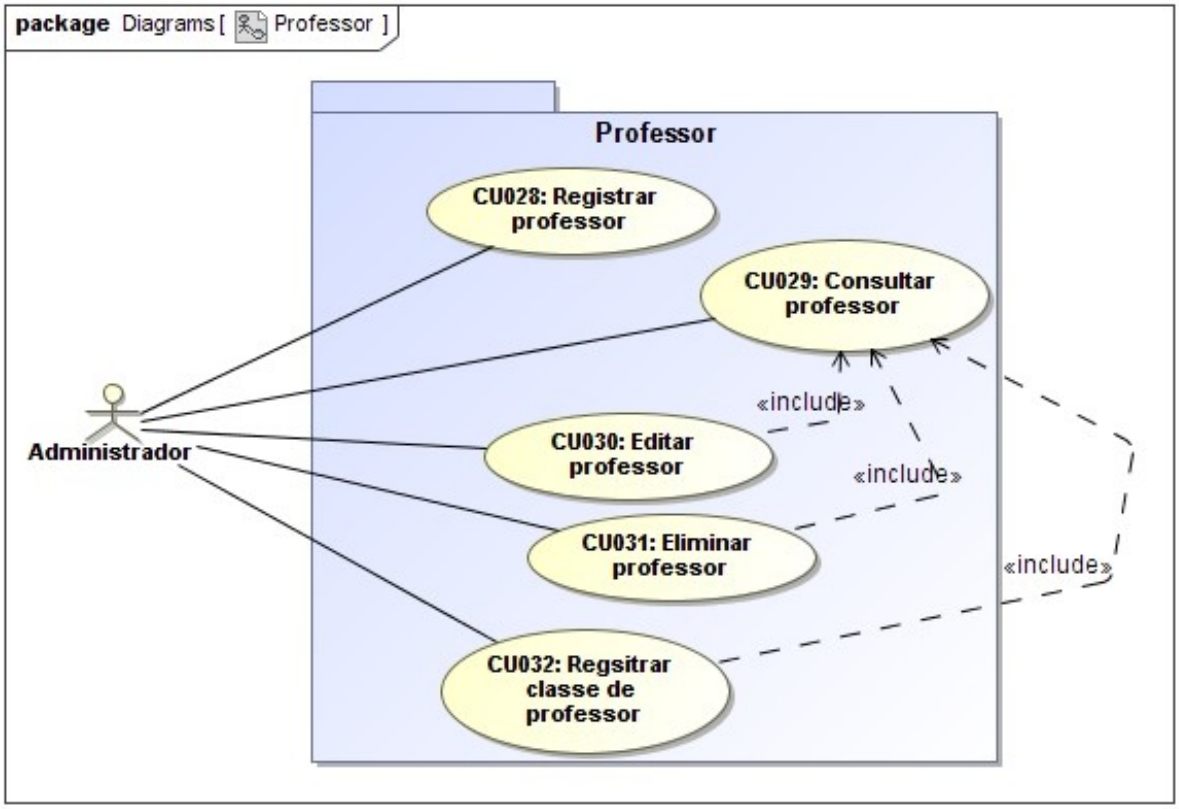


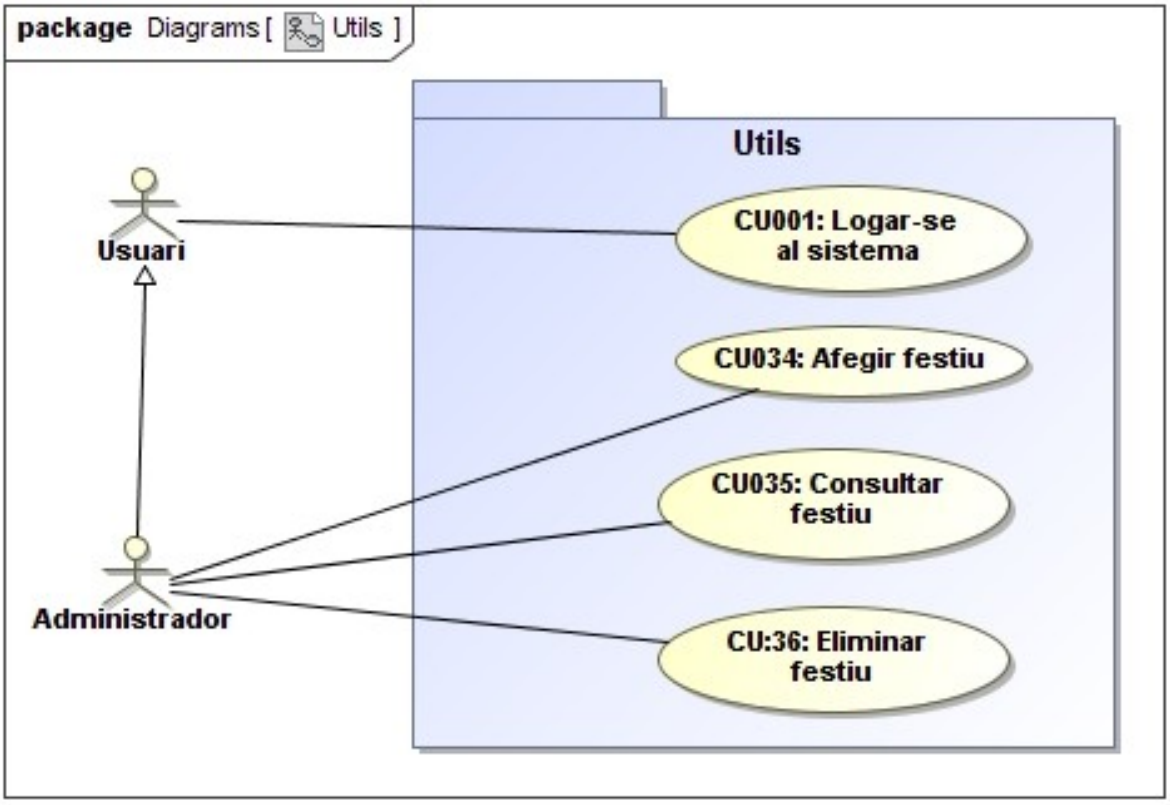

package Diagrams [Grup]
Grup
CU006: Consultar un grup
«include»
CU007: Editar un grup
CU020: Crear un grup
«include»
CU021: Eliminar un grup
Usuari
Administrador


package Diagrams [TipusAbonaments]
TipusAbonaments
CU009: Consultar tipus d'abonament
CU022: Crear un tipus d'abonament
«include»
CU023: Editar un tipus d'abonament
Usuari
Administrador


package Diagrams [Pagament]
CU009: Consultar tipus d'abonament
«include»
CU006: Consultar un grup
Pagament
«include»
Usuari
CU010: Registrar pagament
«include»
CU003: Consultar un alumne
CU011: Consultar pagament
«include»
CU012: Editar pagament
Administrador


package Diagrams [Comptabilitat]
Comptabilitat
CU024: Registrar un despesa
CU025: Consultar despesa
«include»
Administrador
CU026: Editar despesa
«include»
CU027: Eliminar despesa
CU033: Generar informe de comptabilitat


package Diagrams [Professor]
Professor
CU028: Registrar professor
CU029: Consultar professor
«include»
Administrador
CU030: Editar professor
«include»
CU031: Eliminar professor
«include»
CU032: Regsitrar classe de professor


package Diagrams [Utils]
Utils
Usuari
CU001: Logar-se al sistema
CU034: Afegir festiu
CU035: Consultar festiu
Administrador
CU:36: Eliminar festiu

Descripció de casos d'ús

CU001	
Títol	Registrar-se al sistema
Descripció	Entrar al sistema
Actors	Usuari
Precondició	–
Postcondició	L'usuari es troba registrat al sistema amb els permisos que li corresponen
Flux normal	1. L'usuari accedeix al login de l'aplicació 2. El sistema demana les credencials. 3. El sistema comprova les credencials i dóna accés a l'usuari a les funcionalitats que li pertoquen.
Fluxos alternatius	Les credencials no són correctes. 3a. El sistema informa de l'error i el cas d'ús finalitza.

CU002	
Títol	Donar d'alta un alumne
Descripció	L'usuari vol afegir un nou alumne al sistema.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	L'alumne està registrat al sistema.
Flux normal	1. L'usuari introdueix les dades de l'alumne. 2. El sistema comprova que les dades són correctes i guarda l'usuari.
Fluxos alternatius	L'alumne ja existeix. 2a. El sistema informa del problema i s'acaba el cas d'ús. Les dades introduïdes no compleixen les condicions requerides. 2a. El sistema informa dels errors i es torna al punt 3.

CU003	
Títol	Consultar un alumne
Descripció	L'usuari vol consultar la informació d'un alumne.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema mostra la informació de l'alumne.
Flux normal	1. L'usuari introdueix la informació. 2. El sistema cerca l'alumne i en mostra la informació.
Fluxos alternatius	L'alumne no existeix. 2a. El sistema informa del problema i s'acaba el cas d'ús.

CU004	
Títol	Editar un alumne
Descripció	L'usuari vol modificar la informació d'un alumne.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	L'alumne està guardat amb la nova informació.
Flux normal	1. <CU003 Consultar un alumne> 2. L'usuari introdueix les noves dades de l'alumne. 3. El sistema comprova que les dades siguin correctes i guarda la informació de l'usuari.
Fluxos alternatius	Les dades introduïdes no compleixen les condicions requerides. 2b. El sistema informa dels errors i es torna al punt 3.
Dependències	CU003

CU005	
Títol	Consumir classe d'un alumne
Descripció	L'usuari vol registrar que l'alumne ha consumit una classe del seu abonament de grup
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús. L'abonament de l'alumne és de tipus individual.
Postcondició	L'abonament de l'alumne té una classe menys.
Flux normal	1. <CU003 Consultar un alumne> 2. L'usuari introdueix les dades de la classe que es vol consumir. 3. El sistema comprova l'abonament de l'alumne no està exhaurit i es registra que s'ha consumit la classe.
Fluxos alternatius	L'abonament de l'alumne està exhaurit. 3a. El sistema informa dels errors i s'acaba el cas d'ús.
Dependències	CU003

CU006	
Títol	Consultar un grup
Descripció	L'usuari vol consultar la informació d'un grup.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema mostra la informació del grup.
Flux normal	1. L'alumne introdueix les dades del grup o en selecciona un d'una llista. 2. El sistema mostra la informació detallada del grup.
Fluxos alternatius	El grup no existeix. 2a. El sistema informa del problema i s'acaba el cas d'ús.

CU007	
Títol	Editar un grup
Descripció	L'usuari vol modificar la informació d'un grup.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema registra la nova informació del grup del grup.
Flux normal	1. <CU006 Consultar un grup> 2. L'usuari introdueix la nova informació del grup. 3. El sistema comprova que la informació sigui vàlida i la registra.
Fluxos alternatius	La informació introduïda no és vàlida. 3a. El sistema informa dels errors i s'acaba el cas d'ús.
Dependències	CU006

CU008	
Títol	Afegir comentari sobre un alumne
Descripció	L'usuari vol afegir un comentari a la fitxa d'un alumne
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El comentari queda registrat i vinculat amb la fitxa de l'alumne.
Flux normal	1. <CU003 Consultar un alumne> 2. L'usuari introdueix el comentari que vol realitzar. 3. El sistema registra el comentari.
Fluxos alternatius	
Dependències	CU003

CU009	
Títol	Consultar tipus d'abonament
Descripció	L'usuari vol consultar un tipus d'abonament
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema mostra la informació de l'abonament.
Flux normal	1. L'usuari introdueix les dades de l'abonament o en selecciona un. 2. El sistema mostra la informació de l'abonament.
Fluxos alternatius	L'abonament no existeix. 1a. El sistema informa del problema i s'acaba el cas d'ús.

CU010	
Títol	Registrar pagament
Descripció	L'usuari vol registrar el pagament d'un alumne.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El pagament queda registrat al sistema i es mostra la informació de l'abonament.
Flux normal	1. <CU003 Consultar un alumne> 2. <CU009 Consultar tipus d'abonament> 3. <CU006 Consultar un grup> 4. El sistema mostra el preu base a cobrar. 5. L'usuari confirma l'operació. 6. El sistema guarda la informació del pagament, calcula les dates de les classes i mostra la informació detallada del pagament.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-5. El cas d'ús s'acaba. L'usuari vol modificar el preu a cobrar 5a. L'usuari introdueix el nou preu i confirma l'operació.
Dependències	CU003 CU009 CU006

CU011	
Títol	Consultar un pagament
Descripció	L'usuari vol consultar un pagament d'un alumne
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema mostra la informació del pagament.
Flux normal	1. L'usuari introdueix les dades del pagament o el selecciona d'una llista. 2. El sistema mostra la informació del pagament.
Fluxos alternatius	El pagament no existeix. 1a. El sistema informa del problema i s'acaba el cas d'ús.

CU012	
Títol	Editar pagament
Descripció	L'usuari vol modificar el pagament d'un alumne.
Actors	Usuari.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El pagament queda registrat al sistema amb les noves dades i es mostra la informació de l'abonament.
Flux normal	1. <CU011 Consultar un pagament> 2. L'usuari introdueix les noves dades del pagament. 3. El sistema mostra el nou preu final. 4. L'usuari confirma l'operació. 5. El sistema guarda la nova informació del pagament, calcula les dates de les classes i mostra la informació detallada del pagament.
Fluxos alternatius	L'usuari cancel·la l'operació. 2-4. El cas d'ús s'acaba.
Dependències	CU011

CU017	
Títol	Congelar classe d'un alumne
Descripció	L'usuari vol registrar que l'alumne ha consumit una classe del seu abonament
Actors	Administrador.
Precondició	L'usuari està registrat i té rol d'administrador. L'abonament es de tipus grup.
Postcondició	La classe queda congelada i les sessions corresponents a l'abonament queden redistribuïdes.
Flux normal	1. <CU003 Consultar un alumne> 2. L'usuari introdueix la data de la classe que es vol congelar. 3. El sistema comprova l'abonament de l'alumne no està exhaurit, registra que s'ha congelat la classe i recalcula les dates de l'abonament.
Fluxos alternatius	L'abonament de l'alumne està exhaurit. 3a. El sistema informa dels errors i s'acaba el cas d'ús.
Dependències	CU003

CU020	
Títol	Crear un grup.
Descripció	L'usuari vol modificar una sessió.
Actors	Administrador.
Precondició	L'usuari està registrat i té rol d'administrador.
Postcondició	El grup queda guardat al sistema.
Flux normal	1. L'administrador introdueix les dades del grup. 2. El sistema comprova que les dades són correctes i emmagatzema el grup.
Fluxos alternatius	Les dades són incorrectes. 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU021	
Títol	Eliminar un grup.
Descripció	L'usuari vol eliminar un grup.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El grup queda eliminat del sistema.
Flux normal	1. <CU006 Consultar un grup> 2. El sistema demana confirmació per a eliminar el grup. 3. L'usuari confirma que vol eliminar. 4. El sistema elimina el grup.
Fluxos alternatius	L'usuari cancel·la l'operació. 2-4. El cas d'ús s'acaba.
Dependències	CU006

CU022	
Títol	Crear un tipus d'abonament.
Descripció	L'usuari vol crear un nou tipus d'abonament.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El tipus d'abonament queda registat al sistema.
Flux normal	1. L'usuari introdueix les dades del tipus d'abonament. 2. El sistema comprova que les dades són vàlides i guarda el tipus d'abonament.
Fluxos alternatius	Les dades són incorrectes. 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU023	
Títol	Editar un tipus d'abonament.
Descripció	L'usuari vol eliminar un grup.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El tipus d'abonament queda registrat amb les noves dades.
Flux normal	1. <CU009 Consultar tipus d'abonament> 2. L'usuari introdueix les noves dades. 3. El sistema comprova les dades i registra els canvis.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-2. El cas d'ús s'acaba. Les dades són incorrectes. 3a. El sistema informa de l'error i el cas d'ús s'acaba.
Dependències	CU009

CU024	
Títol	Registrar una despesa.
Descripció	L'usuari vol registrar una nova despesa.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	La despesa queda registrada al sistema.
Flux normal	1. L'usuari introdueix les dades de la despesa. 2. El sistema valida que les dades siguin correctes i registra la despesa.
Fluxos alternatius	Les dades són incorrectes. 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU025	
Títol	Consultar una despesa.
Descripció	L'usuari vol consultar una despesa.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El sistema mostra informació sobre la despesa.
Flux normal	1. L'usuari introdueix les dades de la despesa o en tria una d'un llistat. 2. El sistema mostra els detalls de la despesa.
Fluxos alternatius	La despesa no existeix al sistema. 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU026	
Títol	Editar una despesa.
Descripció	L'usuari vol modificar les dades d'una despesa.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	La despesa queda registrada amb les noves dades.
Flux normal	1. <CU025 Consultar una despesa> 2. L'usuari introdueix les noves dades. 3. El sistema comprova les dades i registra els canvis.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-2. El cas d'ús s'acaba. Les dades són incorrectes. 3a. El sistema informa de l'error i el cas d'ús s'acaba.
Dependències	CU025

CU027	
Títol	Eliminar una despesa.
Descripció	L'usuari vol eliminar una despesa.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	La despesa ja no es troba al sistema.
Flux normal	1. <CU025 Consultar una despesa> 2. El sistema demana confirmació. 3. L'usuari confirma que vol eliminar. 4. El sistema elimina la despesa.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-3. El cas d'ús s'acaba.
Dependències	CU025

CU028	
Títol	Registrar professor.
Descripció	L'usuari vol donar d'alta un nou professor al sistema.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El professor queda registrat al sistema.
Flux normal	1. L'administrador introdueix les dades del professor. 2. El sistema valida les dades i registra el nou professor.
Fluxos alternatius	Les dades introduïdes no són correctes: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU029	
Títol	Consultar un professor.
Descripció	L'usuari vol consultar la fitxa d'un professor.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El sistema mostra la fitxa amb les dades del professor.
Flux normal	1. L'administrador introdueix les dades del professor o en tria un d'un llistat. 2. El sistema mostra la fitxa amb els detalls del professor.
Fluxos alternatius	El professor no es troba al sistema: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU030	
Títol	Editar un professor.
Descripció	L'usuari vol modificar les dades d'un professor.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El professor queda registrat amb les noves dades.
Flux normal	1. <CU029 Consultar un professor> 2. L'usuari introdueix les noves dades. 3. El sistema valida les dades i guarda el professor amb les noves dades.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-3. El cas d'ús s'acaba. Les dades introduïdes no són vàlides: 3a. El sistema mostra un missatge d'error i el cas d'ús s'acaba.
Dependències	CU029

CU031	
Títol	Eliminar un professor.
Descripció	L'usuari vol eliminar un professor.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El professor ja no es troba registrat al sistema.
Flux normal	1. <CU029 Consultar un professor> 2. El sistema demana confirmació. 3. L'usuari confirma que vol eliminar. 4. El sistema elimina el professor.
Fluxos alternatius	L'usuari cancel·la l'operació. 1-3. El cas d'ús s'acaba.
Dependències	CU029

CU032	
Títol	Registrar la classe d'un professor.
Descripció	L'usuari vol registrar que un professor ha donat una classe.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El professor té una nova classe registrada.
Flux normal	1. <CU029 Consultar un professor> 2. L'administrador introdueix les dades de la classe. 3. El sistema valida les dades i registra la classe.
Fluxos alternatius	Les dades introduïdes no són vàlides: 3a. El sistema informa de l'error i el cas d'ús s'acaba.
Dependències	CU029

CU033	
Títol	Generar informe de comptabilitat.
Descripció	L'usuari vol generar l'informe de comptabilitat detallat d'un mes.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	El sistema genera un informe de comptabilitat y el mostra per pantalla.
Flux normal	1. L'administrador selecciona el mes del que vol generar l'informe. 2. El sistema genera l'informe i mostra la informació.
Fluxos alternatius	El mes seleccionat és posterior a la data actual: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU034	
Títol	Afegir festiu.
Descripció	L'administrador vol afegir una nova data festiva.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	Hi ha un nou dia festiu registrat al sistema.
Flux normal	1. L'administrador selecciona la data que vol afegir. 2. El sistema guarda la data.
Fluxos alternatius	La data ja estava registrada com a festiva: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU035	
Títol	Consultar festius.
Descripció	L'usuari vol consultar les dates festives registrades.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís per a realitzar el cas d'ús.
Postcondició	El sistema mostra tots els festius.
Flux normal	1. L'usuari indica que vol consultar les dates festives. 2. El sistema mostra el llistat amb les dates festives registrades.
Fluxos alternatius	No hi ha festius registrats: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

CU036	
Títol	Eliminar festiu.
Descripció	L'administrador vol eliminar una data festiva.
Actors	Administrador.
Precondició	L'usuari està registrat i té permís d'administrador.
Postcondició	La data ja no figura com a festiva al sistema.
Flux normal	1. L'administrador selecciona la data que vol eliminar. 2. El elimina guarda la data.
Fluxos alternatius	La data no estava registrada com a festiva: 2a. El sistema informa de l'error i el cas d'ús s'acaba.

Planificació i metodologia

Metodologia de treball

Aprofitant la presència i l'interès del propietari de Corestudio pel projecte es decideix seguir una aproximació a les metodologies àgils, per adaptar el desenvolupament del projecte en funció de seves necessitats. Per poder portar a terme aquesta decisió es plantegen un seguit d'opcions per ser consensuades:

- Disposar, un cop comenci la fase de desenvolupament, d'un **entorn de preproducció** que pugui ser accedit pel client i els professors (així com pel director del projecte) en tot moment per a testejar l'aplicació i fer les indicacions oportunes i fer les correccions necessàries tan aviat com sigui possible per, així, reduir el seu cost. En aquest sentit es decideix desplegar l'aplicació en un servidor remot de la plataforma *OpenShift* on ha d'estar sempre desplegada la darrera versió estable del projecte. Per a assegurar que aquest desplegament es realitza de manera correcta i per a garantir que al servidor sempre hi serà la darrera versió estable del codi que es trobi al repositori s'instal·larà un servidor d'integració contínua *Jenkins*. Aquest Jenkins haurà d'estar connectat al repositori del projecte (que es trobarà a *GitHub*) i haurà de fer un nou desplegament cada cop que es facin canvis a la branca mestra del repositori. Aquest entorn, finalment i en contra de l'equip de desenvolupament, no serà facilitat als professors perquè el client prefereix ser l'única veu autoritzada envers la direcció que ha de seguir el projecte.
- Fer servir una **eina de gestió de projectes** per a monitoritzar l'estat de les tasques. En aquest sentit es proposa la utilització de l'eina *Rally* de gestió de projectes perquè el client pugui estar al cas de l'estat de les tasques i aprovar la seva finalització quan calgui. Aquesta opció va quedar descartada perquè el client va indicar que amb la seva càrrega de feina li seria impossible estar al tant. En conseqüència, la gestió del projecte serà una tasca exclusiva de l'equip de desenvolupament i es portarà de manera privada.
- Mantenir un **flux d'informació** constant, de manera que els dubtes puguin ser aclarits ràpidament i que els desviaments dels resultats esperats pel client siguin trobats aviat per poder corregir-los fàcil i ràpidament. En aquest sentit es planteja l'ús de xarxes socials per al flux de conversació comú, l'ús del correu electrònic per a definicions més formals i reunions puntuals per a aclarir dubtes més complexes o més sensibles per al futur del projecte.

Planificació inicial

Per a poder complir amb els requisits de les entregues parcials pròpies del Treball de Fi de Grau, es planteja una planificació general a mig termini que no obstant, s'anirà corregint a mesura que el projecte avanci per a adaptar-se als canvis que sorgeixin durant tot aquest procés. Així es defineixen quatre grans blocs en la planificació:

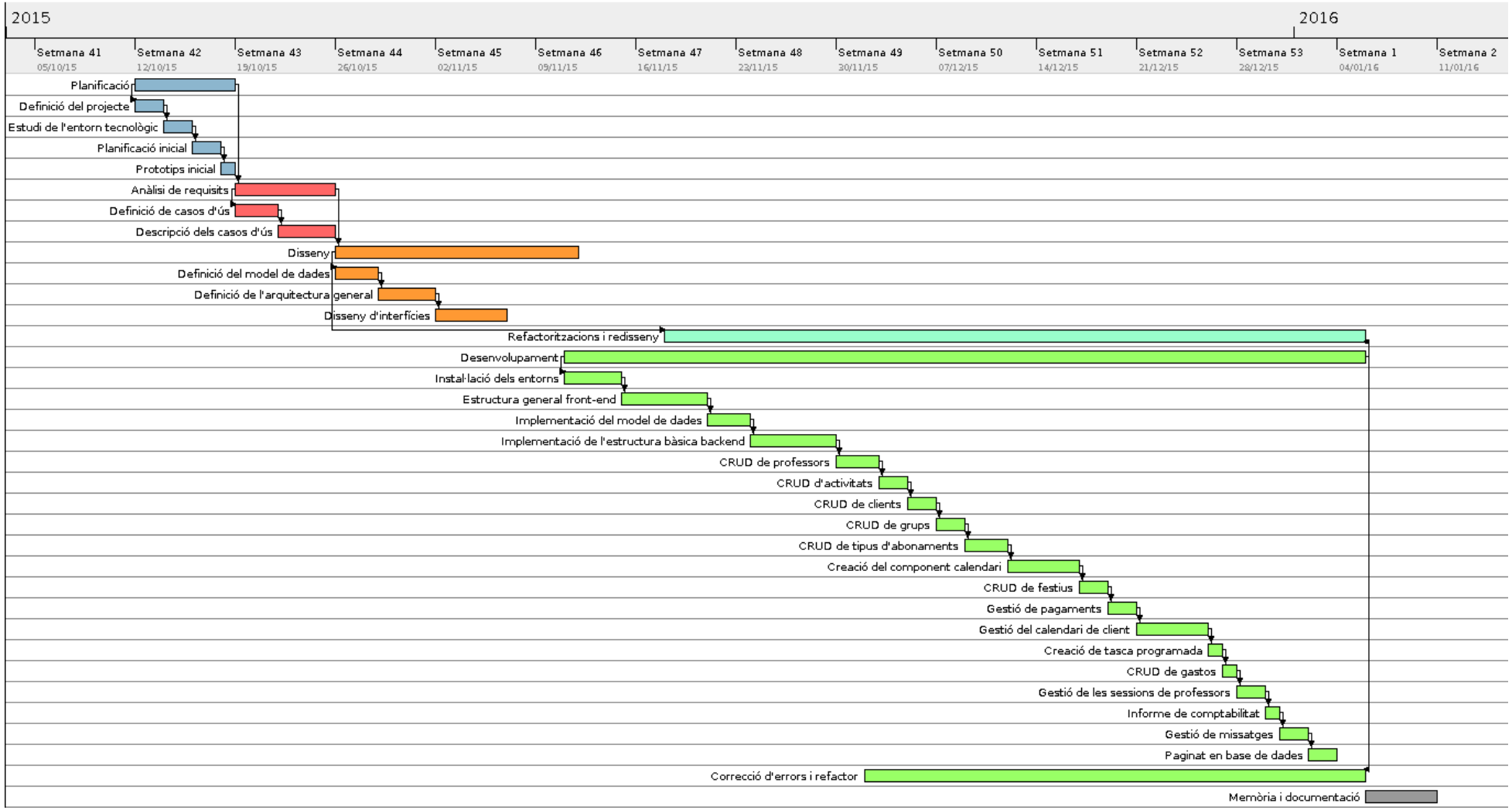
- **Definició del projecte**: que ha de durar les dues primeres setmanes on s'ha de definir la visió del projecte, una primera presa de requeriments i aquesta planificació general.

- **Anàlisi de requisits**: ha de durar dues setmanes i ha de cobrir la definició dels requisits que es plantegen d'inici per al projecte.

- **Anàlisi i disseny**: dues setmanes per a estudiar el problema i decidir l'arquitectura bàsica de la solució al problema i el disseny base de les interfícies d'usuari.

- **Desenvolupament i testeig**: dos mesos durant els quals s'aniran resolent els casos d'ús definits. Durant aquesta fase també s'hauran de fer les correccions que siguin precises tant dels requisits com del disseny plantejat anteriorment. Finalment, durant aquesta fase s'ha de provar que l'aplicació proporciona els resultats esperats.

A continuació s'ofereix un diagrama més detallat de la planificació del projecte.


2015
2016
Setmana 41
05/10/15
Setmana 42
12/10/15
Setmana 43
19/10/15
Setmana 44
26/10/15
Setmana 45
02/11/15
Setmana 46
09/11/15
Setmana 47
16/11/15
Setmana 48
23/11/15
Setmana 49
30/11/15
Setmana 50
07/12/15
Setmana 51
14/12/15
Setmana 52
21/12/15
Setmana 53
28/12/15
Setmana 1
04/01/16
Setmana 2
11/01/16
Planificació
Definició del projecte
Estudi de l'entorn tecnològic
Planificació inicial
Prototips inicial
Anàlisi de requisits
Definició de casos d'ús
Descripció dels casos d'ús
Disseny
Definició del model de dades
Definició de l'arquitectura general
Disseny d'interfícies
Refactoritzacions i redisseny
Desenvolupament
Instal·lació dels entorns
Estructura general front-end
Implementació del model de dades
Implementació de l'estructura bàsica backend
CRUD de professors
CRUD d'activitats
CRUD de clients
CRUD de grups
CRUD de tipus d'abonaments
Creació del component calendari
CRUD de festius
Gestió de pagaments
Gestió del calendari de client
Creació de tasca programada
CRUD de gastos
Gestió de les sessions de professors
Informe de comptabilitat
Gestió de missatges
Paginat en base de dades
Correcció d'errors i refactor
Memòria i documentació

Planificació real

La planificació inicial dels tres primers blocs s'ha complert sense problemes. Durant la fase d'anàlisi i disseny, a més s'ha comptat amb la col·laboració del client durant el disseny de les interfícies d'usuari principals amb l'eina myBalsamiq. D'aquesta manera han sorgit i s'han resolt uns primers dubtes i ha sigut un procés molt enriquidor per al projecte tant per al disseny de les interfícies i com per a refinar els requisits i l'arquitectura de l'aplicació.

Durant la fase de desenvolupament han sorgit més problemes per a mantenir la planificació prevista. D'una banda problemes de caire tècnic, sobretot relacionats amb la seguretat de l'aplicació, d'altra banda la indefinició en quant a la definició dels requisits principals per part del client.

En aquest sentit vam decidir fer una reunió per a redirigir la situació un cop el projecte estava arrencat i el client podia veure, de manera més clara, la forma final de l'aplicació. En aquesta reunió es va fer una nova avaluació de prioritats dels casos d'ús a realitzar per donar més importància a d'altres segons el criteri del client. Així es va decidir bloquejar els casos d'ús de gestionar la seguretat i els accessos a l'aplicació. El client va considerar que, tenint en compte l'aplicació es desplegarà en un servidor d'aplicacions local i sense accés a Internet i que el grau de confiança en els seus treballadors, sumat al fet de que només en són tres (i a més ja tenen accés a gairebé tota la informació de l'empresa), aquesta funcionalitat no és tan important.

A canvi d'aquesta funcionalitat, el client va expressar que li resultaria molt interessant tenir un sistema de missatges que es generessin sota determinades circumstàncies com ara quan els abonaments s'exhaurixen o estiguin a punt. Per tant, es va afegir aquesta funcionalitat sobre els que es van planificar inicialment.

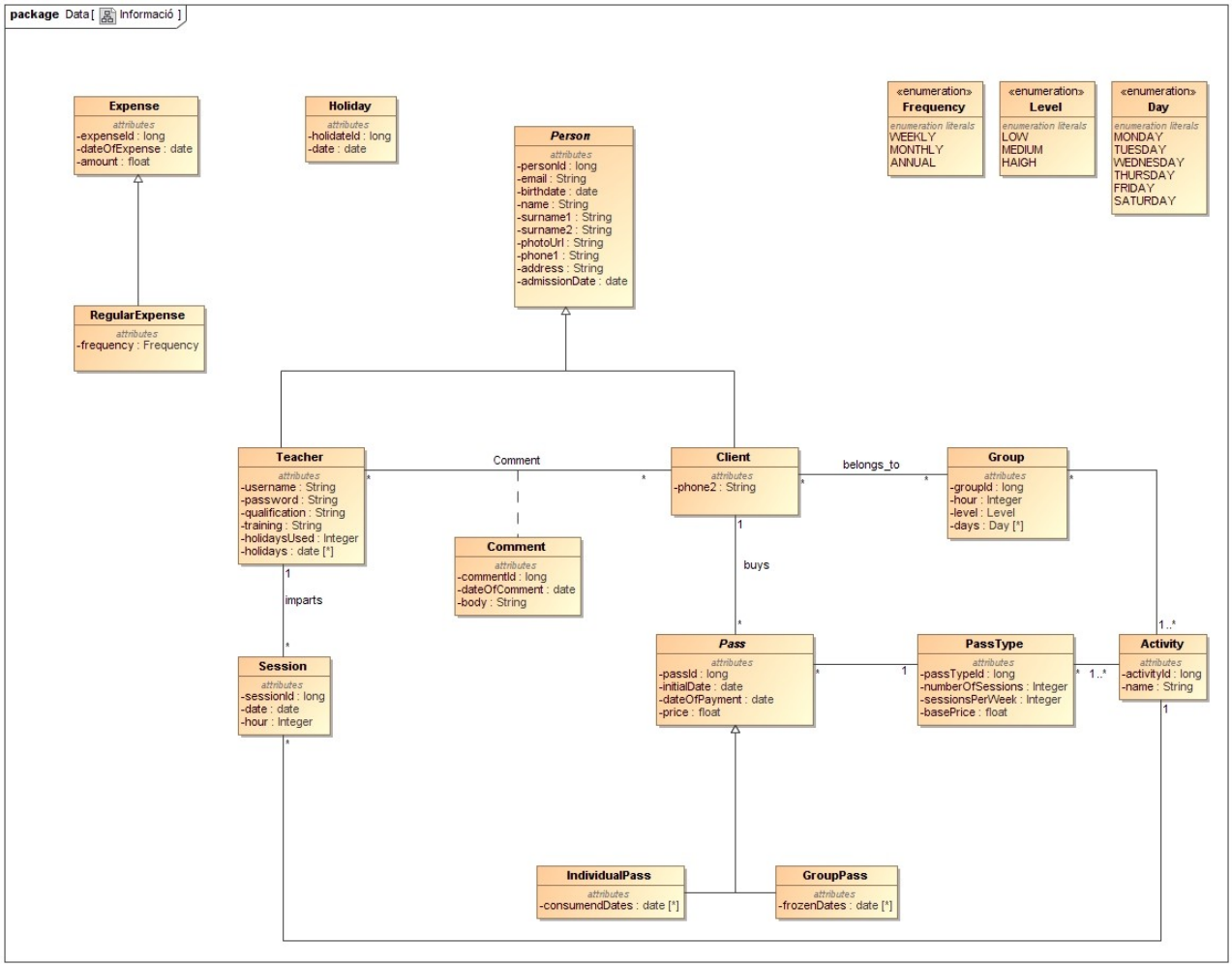
D'aquesta reunió també va sorgir la solució final al punt més conflictiu i important de l'aplicació que és el que té a veure amb el registre dels cobraments i els preus. En el moment d'arrencar el projecte es van definir un seguit de grups de preus, però una anàlisi posterior va descobrir que aquesta definició inicial era incorrecta i que la manera d'atacar el problema que es va decidir durant la fase de disseny era inviable. A més el client va indicar que tenia en ment canviar els trams de preus per a unificar criteris. En aquest sentit es va decidir adoptar una nova solució més flexible perquè el client pugui definir en el moment del cobrament el preu de cada persona sense perdre la informació del preu base de cada tipus d'abonament. També es va decidir eliminar el cas d'ús CU008 perquè el client només el considera una millora addicional sense importància i modificar el cas d'ús i modificar el CU032 perquè només vol registrar les classes que un professor imparteix cada mes de manera global.

A banda d'aquests problemes de caire tècnic, durant la darrera fase del desenvolupament el client ha esta absent del projecte degut a problemes burocràtics. Això va suposar un problema greu perquè durant aquesta fase el volum de funcionalitats resoltes va ser més gran i la seva presència per a fer les proves del sistema per a validar si la solució proposada era l'adequada es va trobar a faltar. Així, durant aquesta darrera fase totes les decisions han recaigut en l'equip de desenvolupament per a poder avançar en el projecte. Aquesta falta de validació i retroalimentació ha provocat un petit descontrol en el procés de desenvolupament i els canvis que s'hi hagin de fer resultaran més costosos.

Anàlisi i disseny

Model de dades

A partir de la primera definició de requisits es va fer una anàlisi del problema i es va fer un esquema inicial de l'arquitectura que hauria de seguir l'aplicació durant el desenvolupament. Aquest disseny inicial suposa una guia no tancada que ha d'evolucionar durant tot el procés de desenvolupament del programari en funció dels canvis sorgits i les refactoritzacions que se'n puguin fer. Així, inicialment es planteja el següent diagrama per a reflectir el punt de vista de la informació, és a dir, el model de dades de l'aplicació.



Amb aquesta primera arquitectura de classes es pretén cobrir els requisits demanats pel client. En funció de la primera definició requisits es va optar per aquesta estructura on el client es relaciona amb **Pass**, que representa un abonament amb una data de començament.

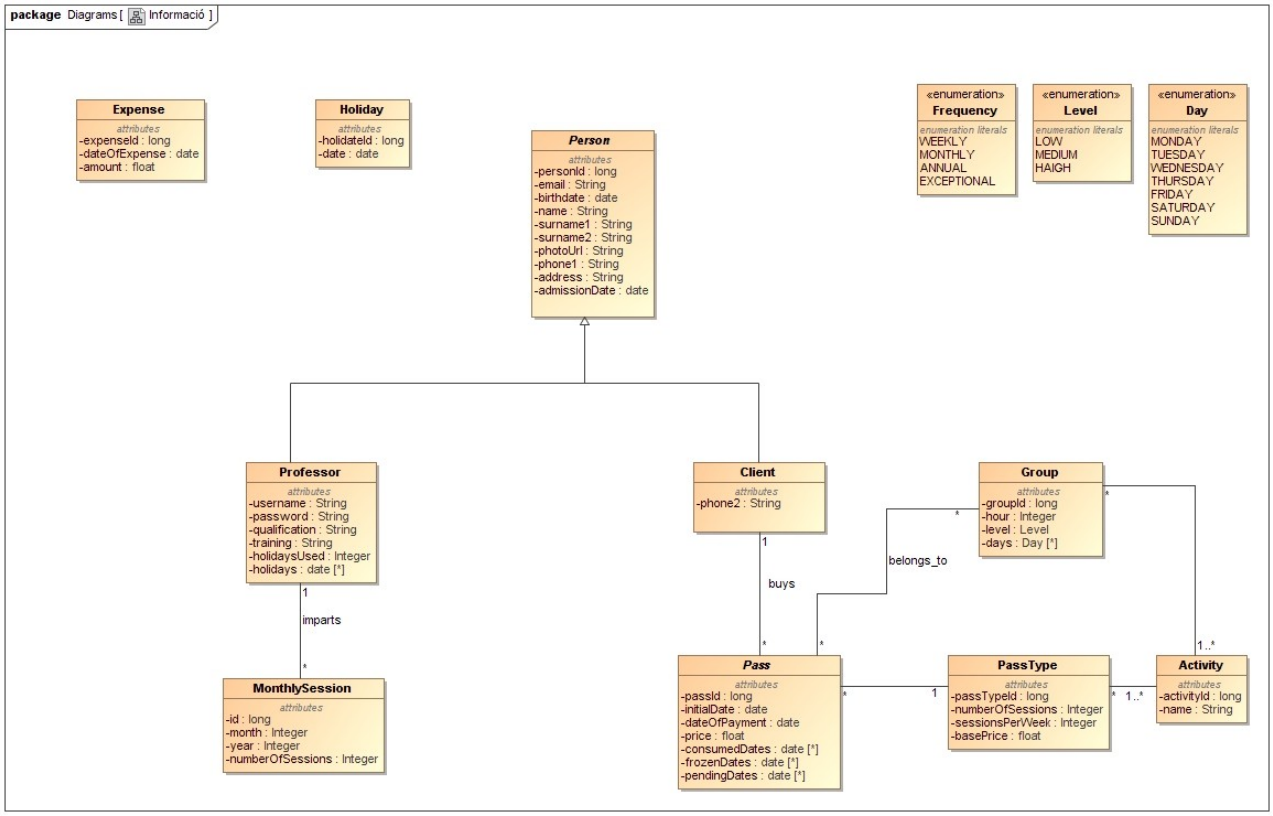
El Pass pot ser **IndividualPass** i **GroupPass** per a atendre les particularitats d'aquests dos conceptes i les seves peculiaritats: la consumició de classes és manual en el **IndividualPass**; es pot congelar classes (comptar com no consumides) en el **GroupPass**. Aquest **Pass** té relació amb un **PassType** que defineix el nombre de sessions i el nombre de sessions setmanal. Amb la informació que proporcionen aquestes dues classes i amb la que proporciona la classe **Holiday**, que representa el llistat de festius definit per l'usuari, podem calcular quan s'esgota un abonament, tenint en compte les peculiaritats descrites anteriorment.

El fet de que el Pass tingui el preu definit serveix per a cobrir la disparitat de preus per a una mateixa activitat que té l'estudi (preus diferents per a clients nous i antics, descomptes del 10% per a determinats clients, preus mínims per a altres clients, etc), el tipus d'abonament té un preu base recomanat però l'aplicació permet introduir-ne un altre. Es va valorar l'opció de fer servir el patró **decorador** per solucionar aquest problema però el responsable de l'estudi ha preferit poder introduir el preu de manera manual millor que de manera preconfigurada.

Durant la fase de desenvolupament, però, s'han introduït petites modificacions sobre aquest esquema inicial:

- Prescindir de les especialitzacions de la classe **Pass** (IndividualPass i GroupPass) i traslladar aquesta informació a l'activitat. Aquesta refactorització ha suposat, a més, la redefinició de la classe Pass que ara inclou tres llistats de dates, les congelades, les pendents i les consumides. D'aquesta manera el sistema permet major flexibilitat en el tractament de dates per part del client i alleugera els càlculs de dates pendents (a costa de requerir més espai en base de dades).
- Canviar l'associació **Client** – **Group** per l'associació **Pass** – **Group**. D'aquesta manera es manté la informació de a quin grup ha estat vinculat un abonament i al mateix temps es pot saber a quin grup pertany un client actualment i, fins i tot, a quins grups ha pertangut.
- Eliminar la classe associativa **Comment** degut a que es va decidir eliminar aquest cas d'ús.
- Eliminar la classe **Session** perquè es va decidir canviar la manera en que s'han d'enregistrar les sessions impartides per un professor. Per a complir el nou requisit s'afegeix una nova classe **MonthlySessions** que conté les sessions que ha impartit un professor per un mes (que està definit pel mes i l'any).
- Eliminar l'especialització de la classe **Expense** i a canvi afegir un valor més a l'enumerat de freqüència que és *Excepcional*.

Així, el diagram final de classes queda com segueix:



Arquitectura del sistema

Per tal de desacoblar el sistema i fer-ho fàcilment extensible i modificable s'ha decidit fer servir una arquitectura en tres capes ben separades:

- **Capa de presentació**: és l'encarregada d'interactuar amb el client. En el cas d'aquest projecte aquesta capa serà la interfície web de l'aplicació. Degut als canvis en la potència dels ordinadors personals i a les possibilitats que proporcionen, tant les noves versions d'html, css i Javascript, com els frameworks disponibles per a fer clients més pesats i amb més lògica, en aquesta capa s'ha emprat el patró Model Vista Controlador.
  - *Model*: està implementat com un seguit de serveis implementats amb AngularJS. Aquesta capa és l'encarregada d'interactuar amb el servidor per a realitzar les peticions necessàries i proveir de les dades al controlador. Per a millorar l'eficiència de l'aplicació, en determinats serveis s'ha implementat una memòria cau que evita peticions redundants al servidor. Aquesta memòria cau s'esborra en fer peticions que canviïn l'estat del servidor (POST, PUT o DELETE) per a mantenir la consistència entre totes dues capes.
  - *Vista*: són les pàgines web amb les que ha d'interactuar el client. Estan implementades fent servir Bootstrap com a framework per a proveir els estils i components bàsics. Per a facilitar i fer més potent i eficient la maquetació i l'escriptura d'estils s'ha decidit fer servir **Sass**. Finalment, la base de l'estructura general de l'aplicació ha sigut un plantilla obtinguda de Start Bootstrap.
  - *Controlador*: està implementat com a controladors de AngularJS. Són els encarregats de controlar les interaccions de l'usuari amb la vista i d'actualitzar les vistes en funció de les dades obtingudes del Model.

- **Capa de negoci**: la capa de negoci també s'ha estructurat en capes per a aïllar la lògica de negoci purament de la part que fa d'interfície o API Rest. Així, tenim:
  - Paquet **rest**: on es troben les els controladors (RestControllers de SpringBoot).
  - Paquet **business**: on es troben les classes que duen a terme la lògica de negoci.

  Aquesta separació permet canviar la forma en la que s'accedeix a l'aplicació (per un servei web o per una aplicació d'escriptori) sense haver de modificar la lògica de

negoci. Dins de cadascun d'aquests paquets s'ha creat una classe que gestiona cadascuna de les entitats descrites a l'aplicació (amb l'afegit de classes que no tenen entitat associada perquè treballen amb dades derivades com les que controlen la comptabilitat). Com que la major part de les operacions són operacions CRUD, hi ha una part que és comuna a quasi bé totes les classes. Per a enfrontar aquest fet s'han creat superclasses que defineixen aquest funcionament comú per a cada paquet:

- Paquet **rest**: hi ha un **BaseRestService** que defineix el comportament de les operacions CRUD i les respostes amb cinc mètodes: *getAllEntities*, *getEntity*, *createEntity*, *updateEntity* i *deleteEntity*. Fent servir un patró **Mètode Plantilla**, aquest quatre mètodes construeixen les respostes cridant el corresponent mètode en la business logic corresponent. Aquesta classe base està tipada fent servir els genèrics de Java. Així, aquesta classe abstracta està tipada de manera **T extends BaseEntity** essent T l'entitat a la que el controlador es correspon per a assegurar que el tipus retornat és el que requerit. El tipus de business logic que es fa servir es decideix en temps d'execució gràcies a un mètode abstracte, getBusinessLogic, que les subclasses han de definir i que retornarà la implementació concreta de *BaseBusinessLogic* que requereix el controlador.

  Com a resposta, aquests mètodes retornen bé una instància de *Page* per al mètode *getAllEntities* o una instància de *ResponseEntity*. En la construcció d'aquesta ResponseEntity afegim una capcelera personalitzada que porta un missatge respecte a l'èxit o l'error de l'operació. Aquest missatge s'ha de construir, també, en funció de la instància de BaseRestService amb un mètode abstracte *getMessage*. Addicionalment, hi ha un altre mètode abstracte, *getUri* que serveix per a obtenir l'uri de la que prové la resposta per a construir la ResponseEntity.

  Amb aquesta estructura, crear un nou controlador només consisteix en crear una classe que estén BaseRestService, definir on ha d'escoltar les peticions (RequestMapping) i implementar els tres mètodes descrits anteriorment. Amb això ja es té totes les operacions CRUD amb molt poc codi. Si algun controlador requereix d'operacions especials, només cal afegir-les sobre la base creada.

- Paquet **business**: paral·lelament, s'ha creat una classe **BaseBusinessLogic**. Aquesta classe duu a terme les operacions necessàries per a realitzar les tasques CRUD. Al igual que *BaseRestService*, aquesta classes està tipada amb l'ús de genèrics per a actuar amb l'entitat concreta corresponent. En aquesta classe tenim els mètodes que ataquen els mètodes descrits per a BaseRestService. Per a poder realitzar la seva tasca, aquesta classe té també un parell de mètodes abstractes que han d'implementar les subclasses. D'una banda tenim un mètode *validateEntity* que es fa servir tant en la creació com en l'actualització d'entitats.

Així, els mètodes de crear i actualitzar entitats també segueixen el patró **Mètode Plantilla** amb una part de l'algoritme que depèn de la implementació concreta que s'està fent servir.

Per a interactuar amb la base de dades es fan servir respositoris de Spring amb interfícies que estenen de **PagingAndSortingRepository**. Així, també existeix un mètode per a obtenir el repositori concret corresponent a la implementació concreta de la business logic.

- **Capa de dades**: la capa de dades està realitzada sobre una base de dades relacional MySQL. Per a facilitar el accés a la base de dades es fa servir SpringData com a implementació addicional de JPA. Les definicions de les taules es fan directament a Java fent servir les anotacions JPA corresponents. Totes les entitats tenen una superclasse **BaseEntity** que defineix l'atribut id de la classe i que serveix com a base per a l'arquitectura descrita abans. També existeix una altra estructura jeràrquica en la relació d'entitats que és la que formen **Person** i les seves subclasses **Client** i **Professor**. Tot i aquesta relació, s'ha decidit fer la relació més senzilla entre les classes, definint Person com a *MappedSuperclass* i, per tant, tenint tots els camps comuns en cadascuna de les taules que tenim de la base de dades. Com que una persona només pot ser client o professor i que en cap cas es farà servir polimorfisme amb aquestes classes, es troba que la solució és adient, atenent a que és la més senzilla d'implementar.

Estructura de l'aplicació

Com s'ha comentat abans, l'aplicació es pot separar en dues aplicacions, l'aplicació de servidor i la de client, i cadascuna té la seva estructura.

Estructura de l'aplicació client

L'aplicació client té, en primer terme dues parts principals, la part de l'aplicació AngularJS i la part de les llibreries i pàgines d'estil. Cadascuna d'aquestes parts es troba en una carpeta concreta del projecte:

- Carpeta **scripts**: és la que conté l'aplicació d'AngularJS. Està, al seu torn, dividida en dues subcarpetes:
  - Carpeta **app**: conté tant les vistes com els controladors de l'arquitectura descrita anteriorment. Cada component de l'aplicació compta amb la seva carpeta on es troben les seves vistes (pàgines html) i els seus controladors (*controllers* d'Angular). Addicionalment, cada component està contingut en un mòdul independent i dins d'aquest mòdul es defineixen les seves rutes amb el mòdul ui-router. Per tant, cada mòdul, que representa un component, té:
    - Un fitxer per a la definició del mòdul, amb les seves dependències si en té i amb la seva funció d'inicialització si en té.
    - Un fitxer per a definir les rutes del mòdul.
    - Un grup de fitxers de vista amb el seu controlador corresponent.

    Tota l'aplicació està dins d'un mòdul principal **corestudioApp** que té com a dependència tots els mòduls que s'han creat i un altre mòdul, **corestudioApp.core** que defineix les dependències externes del projecte (dependències de mòduls de codi obert que s'han utilitzat).

  - Carpeta **component**: que conté, principalment, els serveis (creats com a components *factory* d'Angular) que fan les funcions de model per a interactuar amb el servidor. Cadascun d'aquests serveis es troba dins del mòdul corresponent per a formar una unitat amb els controladors que tenen a veure amb el mateix

component. Addicionalment, en aquesta carpeta es troben la resta de fitxers amb components d'Angular que s'han anat desenvolupant com les directives i els filtres que han sigut necessaris.

- Carpeta **assets**: en aquesta carpeta es troben un seguit de carpetes importats per al projecte:
  - **fonts**: conté les fonts addicionals que es fan servir al projecte. Aquestes fonts són fonts d'icones de les llibreries de bootstrap (glyphicons) i de **Font Awesome**.
  - **images**: són els fitxers d'imatges que es mostren a les vistes.
  - **js**: conté tots els fitxers Javascript de les llibreries que es fan servir al projecte i un fitxer Javascript que s'ha creat per a estendre el prototip d'algunes funcions de Javascript com ara Date o Array.
  - **sass**: conté tots els fitxers sass del projecte, tant els propis, a la carpeta components, com els importats, a la carpeta lib.
  - **styles**: en aquesta carpeta és on es deixa el fitxer custom.css que es fa servir a les pàgines html i que és producte de la compilació de tots els fitxers sass de la carpeta anterior. També inclou els fitxers css de algunes llibreries que s'han fet servir.

Estructura de l'aplicació servidor

La part servidor de l'aplicació està composada d'un paquet principal **com.onewingsoft.corestudio** dins del qual es troben 7 paquets:

- **business**: conté les classes que realitzen la lògica de negoci, tal com s'ha descrit anteriorment.

- **dto**: conté classes de tipus DTO (data transfer object) que es fan servir per a ampliar o reduir la informació que hi ha a les entitats en funcions de les necessitats, a l'hora de fer-la accessible a través de l'api. Això permet reduir la quantitat de dades enviada i al mateix temps incloure informació derivada que no es troba a la base de dades.

- **model**: conté les classes que representen entitats del sistema que persisteixen a la base de dades.

- **repository**: conté les interfícies que serveixen com a punt d'accés a la base de dades, tal com s'ha explicat abans.

- **rest**: conté les classes que fan de controlador i que defineixen l'API pública de l'aplicació de servidor.

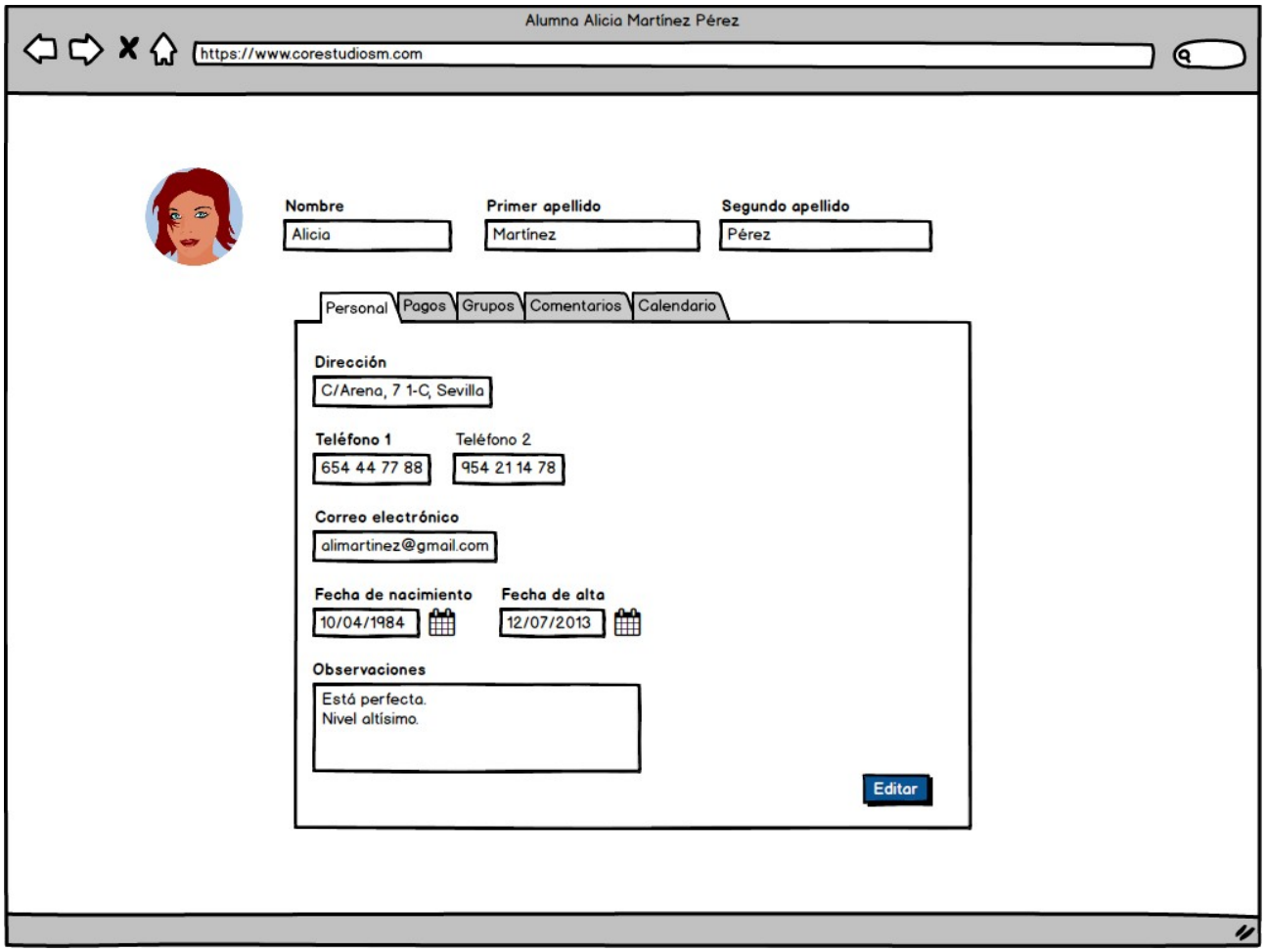
- **security**: conté les classes que gestionen la seguretat del portal. Com que encara no està desenvolupat, actualment està configurat per a permetre l'accés a totes les peticions. Amb tot, conté la configuració bàsica per a implementar la seguretat.

- **utils**: conté classes d'utilitats com són la que permet escriure logs o la que permet crear headers, una classe d'excepció pròpia i els enumerats necessaris a l'aplicació.

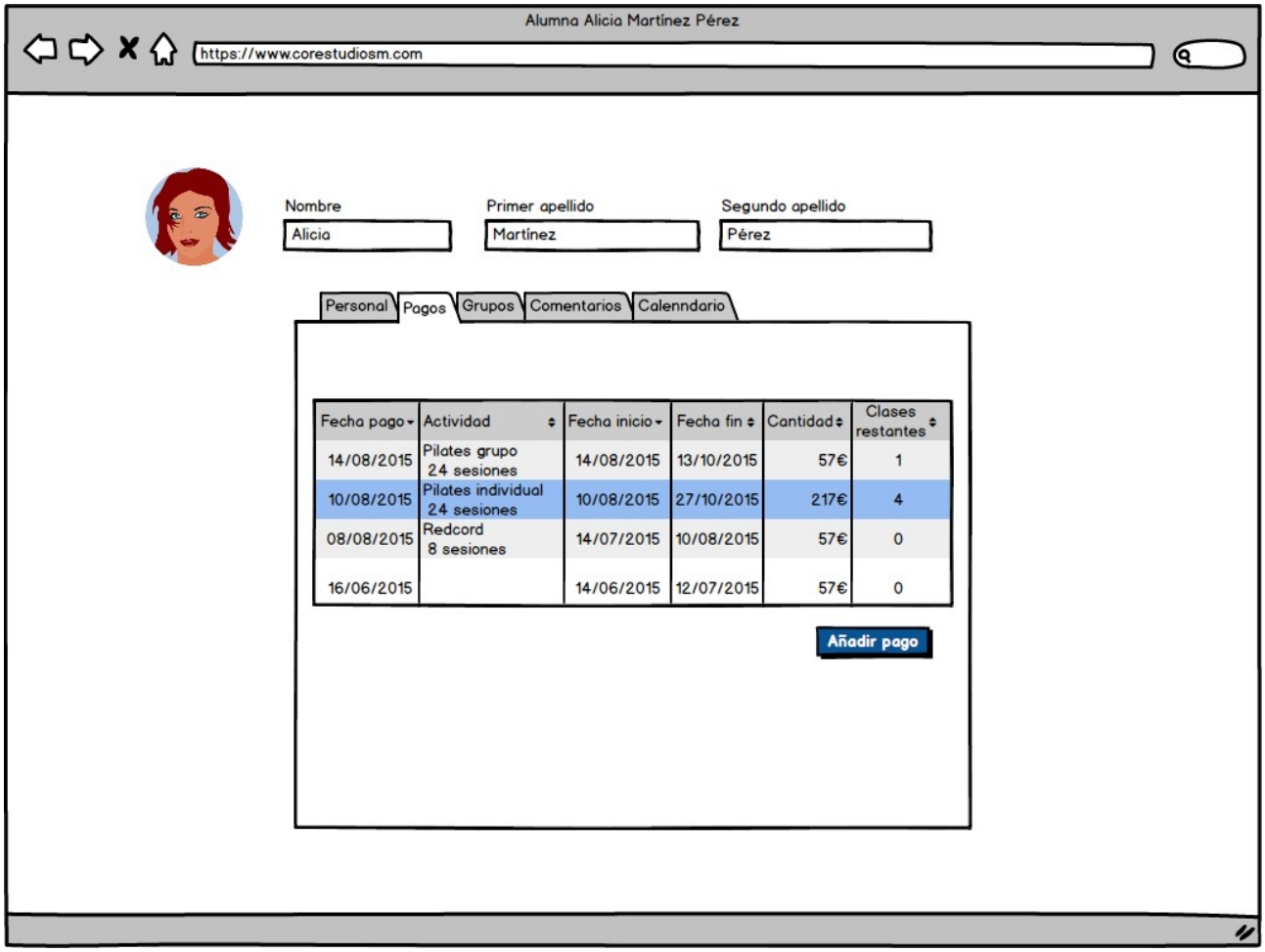
Addicionalment existeix una classe **Applicacion** que serveix per a arrencar l'aplicació de SpringBoot i afegir les configuracions d'inici necessàries, en aquest cas, fixar com a zona horària del servidor l'hora local que es farà servir al client (Europe/Madrid) per tal d'evitar les discrepàncies causades perquè el servidor que es fa servir a l'entorn de proves es troba en una altra zona horària.

Disseny d'interfícies

El disseny de les interfícies va començar amb la creació d'un seguit de wireframes o prototips de les principals pantalles de l'aplicació. Durant aquest procés la interacció amb el client va ser intensa. Segons s'anaven fent, el client sol·licitava millores i canvis. Aquest procés va ser molt enriquidor per al projecte perquè va permetre corregir desviaments i errors respecte als desitjos abans de començar amb la fase de desenvolupament. Els prototips que es van crear van ser els següents:

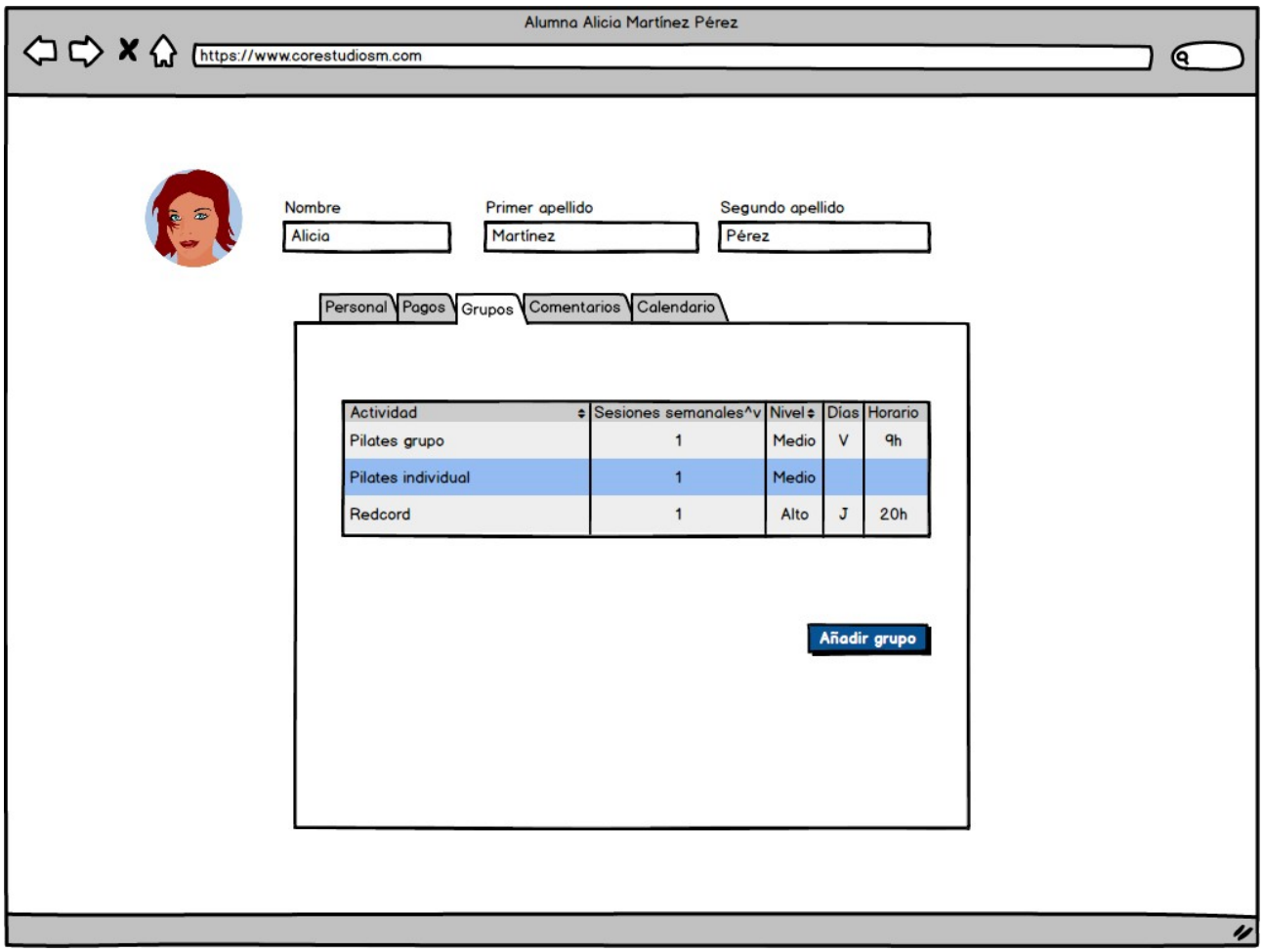


Pantalla de detalls de l'usuari. La pantalla ha de ser compartida o similar a la de creació i edició d'usuari.



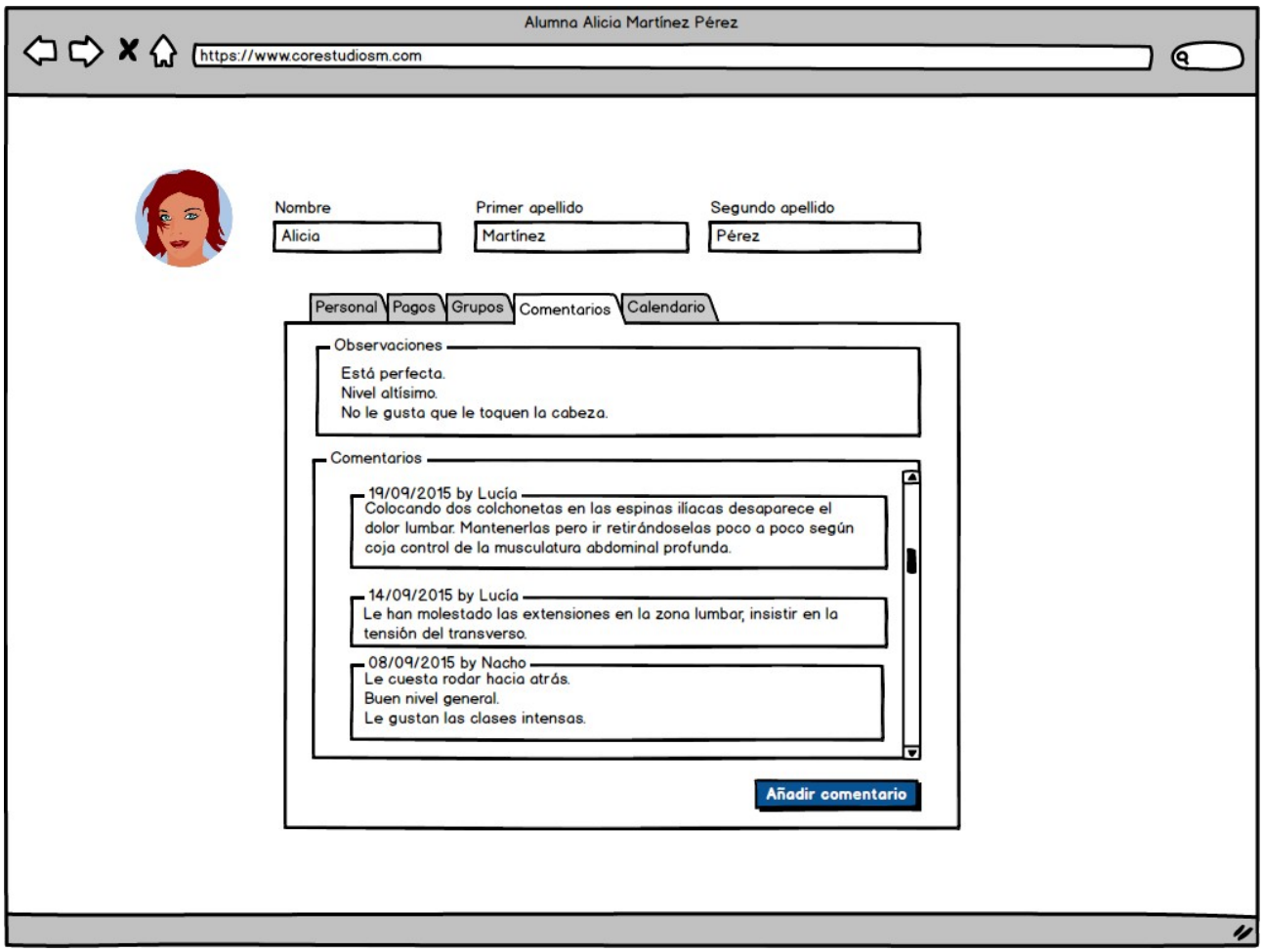
Pantalla amb el llistat de bons del client.

En el disseny d'aquesta pantalla el client va indicar els camps que trobava interessant que es mostressin i l'ordre dels camps. Així, va demanar que sortís la data d'inici de l'abonament i que no calia que estigués pressent la quantitat pagada.



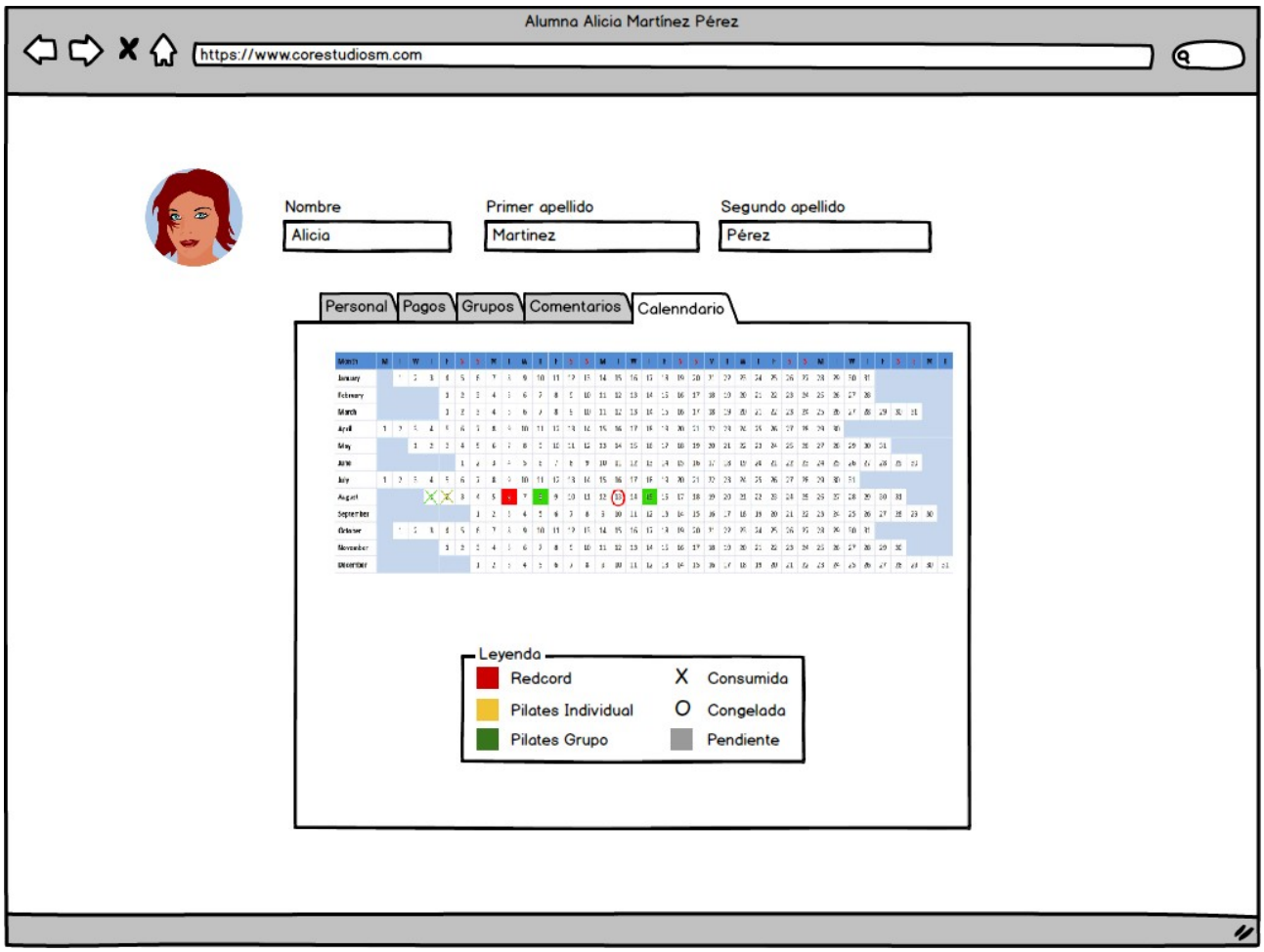
Pantalla de grups del client.

El client va indicar que aquesta pantalla no li oferia una informació gaire important i que era prescindible.



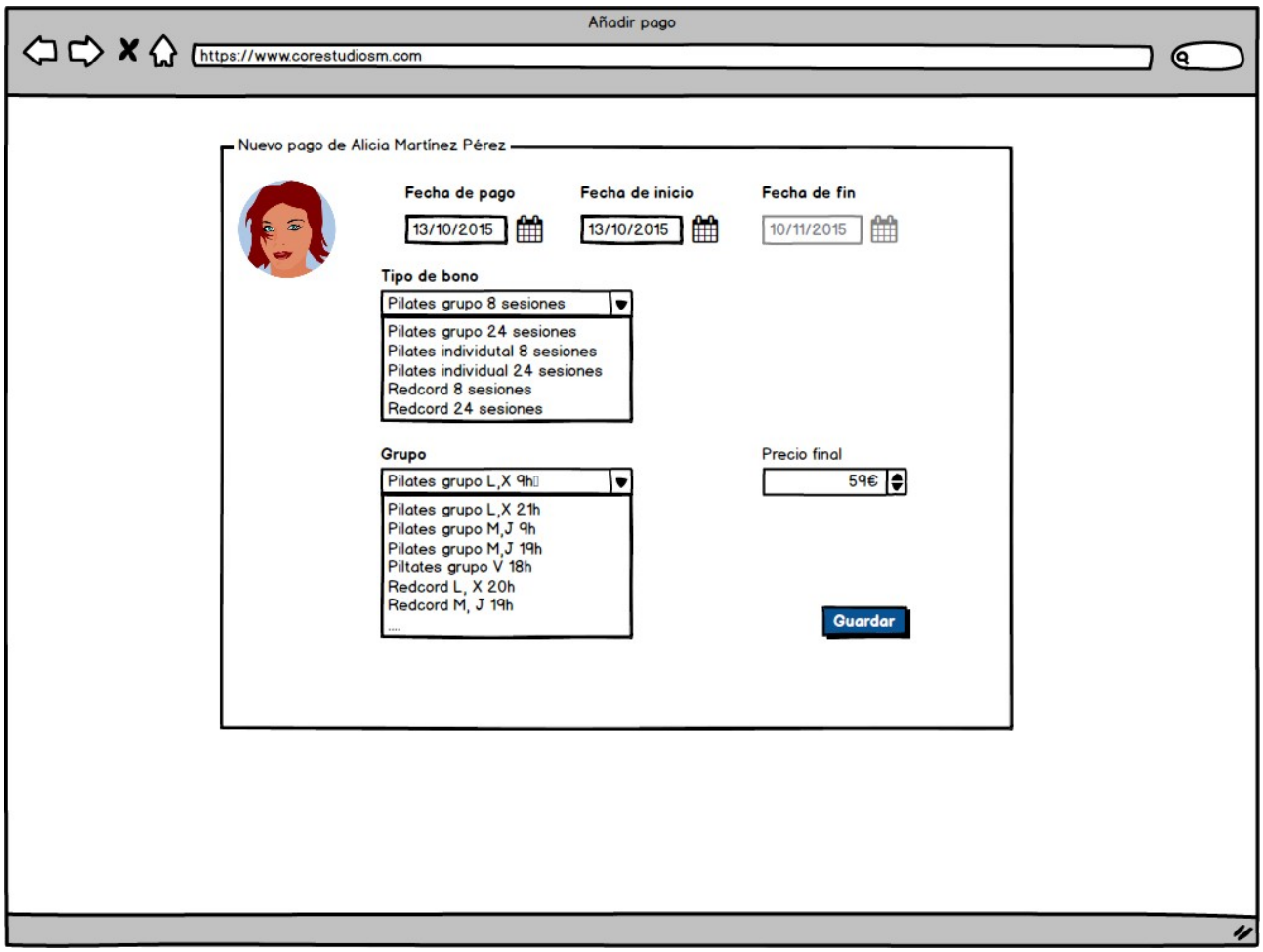
Pantalla de comentaris sobre el client.

Aquesta és una de les funcionalitats que es va eliminar perquè el client no la trobava gaire important en un primer terme.



Pantalla de calendari de sessions del client.

Aquesta és una de les pantalles més importants de l'aplicació. El client va indicar que necessitava que el calendari es mostrés de manera horitzontal i que mostrés la informació de tot l'any en contra d'un calendari tradicional que només mostrés la informació d'un més.



Pantalla de creació d'un abonament.

Aquesta és una altra de les pantalles més importants de l'aplicació i la que més canvis ha patit durant el desenvolupament del prototip. El client volia tenir més flexibilitat i llibertat a l'hora de decidir el preu de l'abonament.

Implementació i stack tecnològic

Client

Stack tecnològic

A la part client s'han fet servir un bon nombre de tecnologies, la majoria de les quals són estàndards en el treball en front-end.

- **Sass (*Syntactically Awesome StyleSheets*)**: és un metallenguatge que permet facilitar i fer més llegible els fitxers d'estils d'una pàgina web. És una extensió de CSS3 que afegeix niuar regles, fer servir variables, crear mixins. És un llenguatge compilat a CSS que és el que, finalment, s'afegeix al fitxer html. Aquesta tecnologia (o la seva alternativa, ***Less***) és bàsica per a crear els estils d'un lloc web perquè facilita molt el procés. En el cas concret d'aquest projecte s'ha aprofitat, especialment:

  - La possibilitat de crear una estructura de fitxers d'estils per components que, finalment, són compilats en un únic fitxer CSS que és el que s'importa al projecte.
  - La definició d'unes constants comunes que representen la imatge del lloc i que són fàcilment accessibles gràcies a la possibilitat de crear constants. La base, en aquest cas, ha sigut les variables definides a la plantilla de *Bootstrap* que s'ha fet servir, sobreescrivint-les quan era necessari.

- **Gulp**: és una eina d'execució de tasques feta amb Javascript i construïda sobre Node.js. Permet l'execució de tasques i l'enllaçat entre tasques (streaming) fent servir (de manera similar a com ho fan els *pipes* d'Unix) la sortida d'una tasca com l'entrada de la següent. Disposa de multitud de mòduls per a executar tasques comunes. En el marc d'aquest projecte s'ha fet servir, principalment per a realitzar la compilació dels fitxers SASS i per a muntar un servidor de desenvolupament que actualitza automàticament la pàgina en detectar canvis en els fitxers que s'hagin indicat, d'aquesta manera, els canvis es reflecteixen immediatament en el navegador accelerant enormement el procés de maquetat.

- **npm**: és el gestor de paquets per defecte de Node.js. Permet la instal·lació de mòduls de Node.js com són Bower o Gulp i tots els seus mòduls. A més, permet mantenir actualitzats tots els mòduls o mantenir-ne una versió concreta. Addicionalment, permet traslladar la configuració del projecte sense haver de moure els mòduls, ja que l'estructura de mòduls queda registrada en un fitxer JSON del que npm llegeix quan s'indica l'ordre *npm install,* per a cercar i instal·lar els mòduls a les versions requerides.

- **Bower**: és el gestor de paquets més utilitzat al desenvolupament del costat client de pàgines web. Depèn de Node.js i npm i treballa amb GitHub per a gestionar les dependències d'un projecte. De manera semblant a com ho fa *npm*, la informació dels paquets instal·lats i les seves versions queda emmagatzemada en un fitxer JSON amb les versions requerides. Igualment, permet fer una instal·lació, a partir d'aquest fitxer, de totes les dependències del projecte.

- **Bootstrap**: és un framework de codi obert, desenvolupat per *Twitter,* per al disseny i maquetat d'aplicacions web. Ofereix un seguit de classes CSS i funcionalitats Javascript per a facilitar el procés de creació de pàgines web. Addicionalment, ofereix suport per a crear llocs *responsive*. Permet l'extensió i l'adaptació dels estils proveïts ja sigui amb *Less* o amb *SASS* i existeixen un gran nombre de plantilles i temes disponibles. En el marc d'aquest projecte s'ha fet servir de manera intensiva, sobretot per a definir l'estructura de les pàgines (fent servir el seu grid). El tema i l'estructura principal provenen d'un tema disponible a Start Bootstrap que s'ha adaptat lleugerament.

- **Font Awesome**: és una llibreria d'icones, representades com a fonts, per a fer servir a les pàgines web. Està basada en CSS i LESS i permet incorporar les icones amb classes CSS. En tractarse de fitxers de fonts, es permet totes les accions que es poden fer sobre les fonts de manera molt senzilla i amb pocs problemes de rendiment o qualitat, com són el canvi de mida o color.

- **AngularJS**: és un framework de Javascript de codi obert desenvolupat per *Google*, per a la creació d'aplicacions web d'una sola pàgina (*Single Page Aplication*), per a millorar l'experiència d'usuari en les aplicacions web riques (*Rich Internet Applications*). Facilita i estructura el procés de creació d'aquest tipus d'aplicacions

gràcies a que aporta una arquitectura a l'aplicació. Dóna suport a arquitectures MVC (Model View Controller) i MVVM (Model View View-Model) i per tant es diu que és un framework MVW (Model Vista Whatever). Es basa, d'una banda, en l'extensió del llenguatge html mitjançant la creació i l'ús de directives per a crear components i, d'altra banda, en el *Two-way data binding*, que permet mantenir consistència entre el que és a la pàgina html i el que succeeix al controlador d'Angular corresponent de manera molt senzilla. Disposa de gran quantitat de mòduls (propis de la plataforma o proveïts per la comunitat) per a estendre les seves funcionalitats.

Implementació

Per a dur a terme aquesta aplicació s'han fet servir un seguit de mòduls d'Angular i també s'han hagut de crear alguns components. Els mòduls que s'han fet servir són:

- **ui-router**: per a gestionar l'encaminament de l'aplicació. Ofereix funcionalitats addicionals sobre el mòdul per defecte (ngRouter) com és la possibilitat de niuar vistes. Com s'ha explicat abans, tot l'encaminament de l'aplicació es gestiona al costat client i aquest és el mòdul que s'encarrega d'aquesta tasca.

- **ngResource**: és el mòdul encarregat de gestionar les peticions al servidor. Permet definir un seguit de mètodes respecte un recurs concret (endpoint). En el cas d'aquesta aplicació permet crear mètodes per a les aplicacions CRUD bàsiques. Permet, a més, la creació de mètodes propis, afegits a partir del recurs inicial. Facilita i fa més neta aquesta tasca. En el projecte s'utilitza en els serveis per a formar un component que fa d'API intermediària entre el controlador i el servidor clarificant molt el codi.

- **smart-table**: és un grup de directives per a la manipulació de taules. Facilita tasques com ara l'ordenació o el paginat.

- **angular-bootstrap-show-errors**: és una directiva que permet mostrar errors de validació sobre els formularis. Aquesta directiva s'ha modificat perquè s'acomodés a les necessitats del projecte, estenent la seva funcionalitat amb una subdirectiva.

- **angular-messages**: és una directiva que facilita mostrar missatges d'errors de validació en els formularis.

- **angular-bootstrap**: és un mòdul que ofereix un grup de directives per a fer servir amb components de bootstrap afegint-ne funcionalitat. En el cas de l'aplicació s'ha fet servir per a mostrar els calendaris en els camps de data dels formularis, per a gestionar les finestres modals i per a gestionar els panels de navegació (nav tabs).

- **angular-local_es-es**: és un mòdul que permet que Angular faci servir els locale correctes per a filtres com el de moneda (currency).

- **angular-animate**: mòdul que permet l'animació en components.

- **angular-treasure-overlay-spinner**: directiva que vela una part de la pàgina i mostra una icona per a indicar que una operació està tenint lloc.

- **checklist-model**: directiva per a associar un grup de valors de camps d'entrada de tipus checkbox a una mateixa variable.

Tot i totes aquestes facilitats afegides al projecte, durant el procés de desenvolupament ha sorgit la necessitat de crear-ne algunes específiques per al projecte.

- **passType**: filtre que a partir d'un objecte tipus PassType, retorna una cadena de text que representa aquest tipus d'abonament (nom de l'activitat i nombre de sessions).

- **horitzontalCalendar**: és una directiva creada per a crear i gestionar calendaris horitzontals. La directiva és un element que estén html dibuixant un calendari en format horitzontal a partir de l'any indicat. Permet, a més, definir un seguit de dates seleccionades (amb un tipus, que representa una classe) per a ser marcades al calendari. També permet definir una funció com a funció que s'ha de crida en fer click sobre una data.

- **siNo**: filtre per a traduir un booleà a cadena de text fent el codi més net i senzill a l'arxiu html corresponent.

- **translateDays**: filtre que, a partir d'un nom de dia, tal com els proveeix el back-end (en majúscules i en anglès), retorna la seva traducció a castellà i en minúscules.

- **translateLevel**: filtre que, a partir d'un nom de nivell, tal com els proveeix el back-end

(en majúscules i en anglès), retorna la seva traducció a castellà i en minúscules.

- **toUderscore**: filtre que substitueix els espais per guionets baixos. Es fa servir per crear ids d'elements html de manera dinàmica en funció de l'entrada i fer servir, així, funcionalitats de Bootstrap que en depenen.

- **range**: filtre que rep un array buit i l'omple amb els valors que hi ha entre el mínim i el màxim indicats.

- **hour**: filtre que afegeix ':00' a l'entrada. Es fa servir per a mostrar les hores en el selector que serveix per a crear grups.

- **daysArray**: filtre que transforma un array de dies en el format del servidor (en majúscules i en anglès) en un array de diminutius dels dies en minúscules i en castellà.

- **replaceAccents** i **permalink**: filtres que eliminen els accents, el primer i que crea una cadena composta pel nom i el primer cognom d'una persona separats per un guionet. Es fan servir de manera conjunta per a crear els permalinks, que són cadenes de text que s'afegeixen a l'url que es mostra al navegador per fer-la més entenedora.

- **group**: filtre que a partir d'un grup crea una cadena de text que el representa.

- **matchValidator**: directiva que permet indicar un camp contra el que s'ha de comparar el valor del que la conté per a comprovar que són iguals. Es fa servir per a la validació de contrasenyes.

- **utils.js**: fitxer de Javascript on s'estenen algunes funcionalitats de Javascript:
  - **isSameDay**: afegida al prototip de Date, compara la data amb altra tenint només en compte el dia, el mes i l'any.
  - **toJSON**: afegida al prototip de Date, posa l'hora a 0 abans de convertir la data a JSON perquè a l'aplicació les hores no es tenen en consideració.

- **getMonthName**: afegida al prototip de Date, retorna el nom del mes en castellà.
- **indexOfId**: afegida a Array, retorna la posició en l'array de l'element amb l'id indicat o -1 si no s'hi troba.

Servidor

Stack tecnològic

A la part servidor el nombre de tecnologies és molt més reduït:

- **Maven**: és una eina per a la gestió i la construcció de projectes Java. Permet definir tant les dependències com els descriptors de desplegament amb un fitxer de configuració (pom.xml), a partir del qual gestiona i descarrega les dependències indicades. A més gestiona el cicle de vida de l'aplicació definint-ne cinc parts: compilació, testeig, empaquetat, instal·lació i desplegament. Addicionalment, permet crear projectes a partir d'arquetips predefinits i, fins i tot, adaptar un projecte ja creat per a ser utilitzat en un IDE concret.

- **Wildfly**: és un servidor d'aplicacions Java EE de codi obert. Està construït amb Java, cosa que permet instal·lar-lo en qualsevol sistema operatiu. Es configura amb un fitxer xml (standalone.xml). És el mateix que proveeix OpenShift (plataforma al núvol on s'allotjat l'aplicació com a entorn de preproducció).

- **SpringBoot**: és una extensió de Spring que facilita i redueix el procés de configuració de l'aplicació i permet crear aplicacions autònomes. Es basa en la convenció sobre la configuració i fa servir anotacions sobre fitxers xml per a configurar l'aplicació. Té un gran nombre de mòduls per a ampliar la seva funcionalitat. En el marc d'aquest projecte s'han fet servir:
  - **Data JPA**: per a gestionar la interacció amb la base de dades.
  - **Security**: per a configurar la seguretat de l'aplicació.

Implementació

A part de les decisions descrites a l'apartat de l'arquitectura de l'aplicació, durant el desenvolupament del projecte s'han pres algunes decisions d'implementació que cal destacar:

- **Maneig dels tipus moneda**: treballar amb quantitats monetàries pot ser complexe per la manera en que Java treballa amb els decimals. Existeixen tipus que en faciliten el seu control com és la classe BigDecimal. Aquesta classe, però, fa molt verboses operacions bàsiques com són la suma i la resta. És per això que s'ha decidit per una altra solució en aquest projecte. Les quantitats s'emmagatzemen en base de dades com a nombres sencers (Integer) i totes les operacions que es fan sobre les quantitats es fan mantenint aquest tipus de dades. Per a les comunicacions amb l'API pública, però, s'han d'acceptar i enviar els valors reals amb decimals. És per això que les entitats o classes que tenen valors econòmics tenen aquests valors marcats amb l'etiqueta **@JsonIgnore** que evita que siguin serialitzats i deserialitzats al convertir l'objecte a o des de JSON. Addicionalment disposen d'uns getters i setters que treballen amb el tipus **Double** que s'encarreguen de convertir, en un sentit i en l'altre, els valors emmagatzemats de la manera correcta (dividint o multiplicant per 100).

- **Tasca programada**: una de les parts que més temps ocupa al client és el control de l'assistència dels clients i la revisió de les sessions pendents dels abonaments. És per això que era de vital importància que això fos gestionat de manera automàtica. Per complir aquest requisit s'ha creat una tasca programada (amb l'anotació de SpringBoot **@ScheduledTask** i habilitant que es puguin fer servir amb l'anotació **@EnableScheduling**) que cerca tots els abonaments que tenen sessió pendent el dia en curs i fa el procés de consumir la sessió. Addicionalment, es genera un missatge si el nombre de sessions restants és menor o igual a dos. La tasca s'ha programat per que es llenci cada sis hores de manera que, per interrupcions en el servidor, no quedi un dia sense processar.

Dades

Com a base de dades s'ha optat per una base de dades relacional MySQL. Després d'analitzar l'alternativa que representen les bases de dades no relacionals, s'ha decidit fer servir una base de dades relacional degut a que les necessitats del projecte no justifiquen l'ús de bases de dades no relacionals. És un producte que ha de treballar de manera local amb un sol client, amb la qual cosa, els beneficis que ofereixen aquestes bases de dades perden tot el sentit. A més, les dades a emmagatzemar tenen una estructura molt concreta i es produiran un nombre similar de consultes i escriptures en la base de dades, per la qual cosa s'ha considerat que és una millor solució una base de dades relacional.

Durant la fase de desenvolupament, principalment s'ha accedit a la base de dades amb el client de consola que ofereix MySQL tot i que, puntualment s'ha fet servir Wokbench com a eina de suport.

Entorn

A part de les eines purament de programació, s'han fet servir altres eines molt importants durant el projecte:

- **GitHub**: com a sistema de control de versions s'ha fet servir **git** i s'ha allotjat el projecte en un servidor privat de **GitHub** (gràcies a un compte d'estudiant). Git és un sistema de control de versions distribuït centrat en la velocitat i la integritat de les dades. Com que és un sistema distribuït, cada directori de git disposa del repositori complet i tota la seva història. Addicionalment, facilita molt la gestió de branques. En el marc d'aquest projecte s'han fet servir branques per funcionalitats.

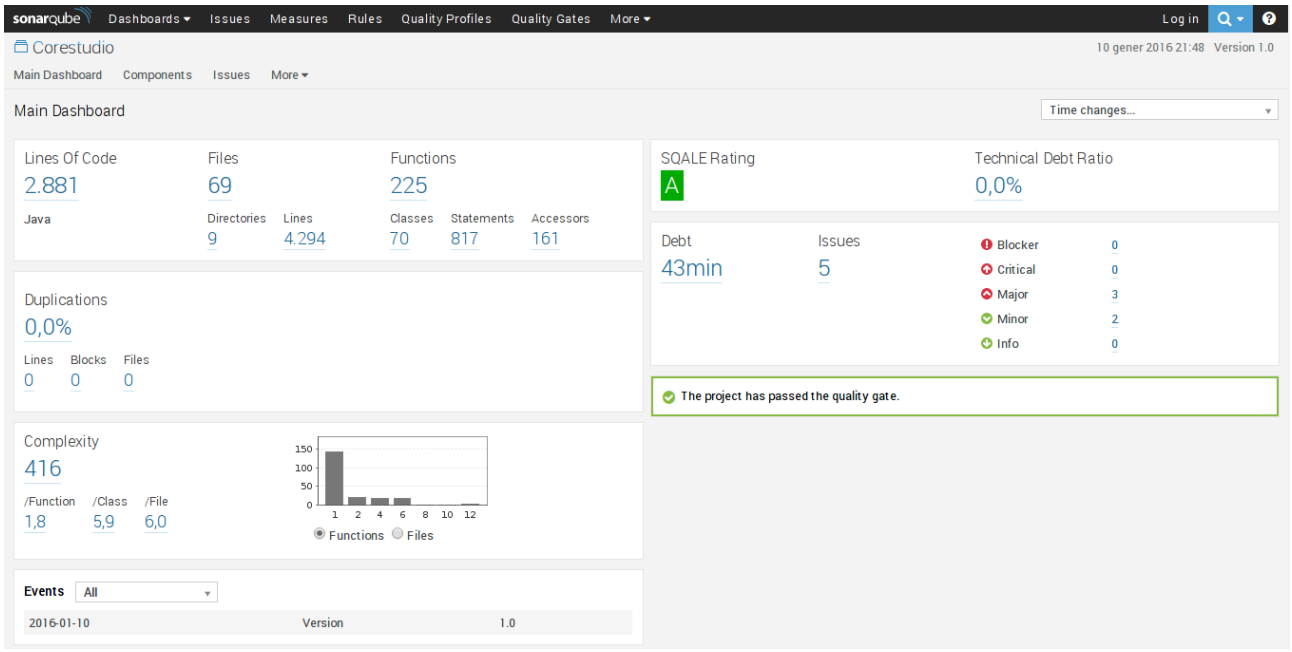
- **OpenShift**: és un producte de plataforma com a servei que permet desplegar projectes instal·lant diferents cartutxos. En aquest projecte s'ha fet servir com a entorn de preproducció que estigués disponible en tot moment per al client i el consultor de l'assignatura. En aquesta plataforma s'ha pogut replicar amb molta fidelitat l'entorn de producció instal·lant-ne un servidor d'aplicacions WildFly i una base de dades MySQL.

- **Jenkins**: és un programari de codi obert basat en el projecte Hudson que serveix per a proporcionar integració contínua en el desenvolupament. En el marc d'aquest projecte, s'ha fet servir per a facilitar i automatitzar els desplegaments en l'entorn de preproducció. Es va instal·lar com un cartutx addicional a la plataforma OpenShift i es va connectar amb el repositori de GitHub de manera que amb cada push sobre la branca mestra es llancés un nou desplegament amb el nou codi.

Build de Jenkins llançat automàticament per un push a la branca mestra del repositori.

Informació de Jenkins sobre els desplegaments del projecte.

- **SonarQube**: és una plataforma de codi obert pera a la inspecció contínua de codi. Ofereix mètriques de qualitat del programari atenent a les convencions pròpies del llenguatge. En aquest projecte s'ha fet servir per a inspeccionar el codi Java. S'ha treballat durant tot el projecte per a mantenir una qualificació A i per a tenir tant poques marques d'errors com sigui possible.



Darrer report de **SonarQube**.

Avaluació de costosos

Al començament del projecte es va estimar una dedicació d'unes 20 hores setmanals. El total del projecte, des del començament formal de l'assignatura ha sigut de 12 setmanes. No obstant, el procés d'elaboració del projecte va començar abans amb algunes reunions per a la toma de requisits i el procés subsegüent de definició d'aquests requisits. És per això que els costos obtinguts de l'elaboració del projecte són:

	Hores	Preu per hora	Cost
Analista	50	35 €	1750 €
Dissenyador	50	25 €	1250 €
Desenvolupador	170	15 €	2550 €
Total	**270**		**5550 €**

Futur de l'aplicació

Amb la conclusió d'aquesta fase del projecte (la que es correspon amb el treball Fi de Grau), es considera que es lliura un producte mínim viable. És un producte amb el que el client ja podrà fer les tasque que ell ha considerat bàsiques durant el procés de desenvolupament. Amb aquest producte el client podrà tenir una control exhaustiu de la comptabilitat del seu negoci, de manera centralitzada, automatitzada i més senzilla i li hauria de facilitar la seva tasca, que és el objectiu del projecte.

Tot i que el producte es considera un producte en sí mateix, que pot entrar en producció, hi ha un bon nombre de tasques que han quedat al backlog del projecte i que es poden dur a terme si el client així ho desitja:

- Adaptar l'aplicació perquè es vegi correctament a un dispositiu mòbil tipus tauleta (iPad). En previsió d'aquesta circumstància, s'ha fet servir Bootstrap que facilita aquesta conversió.

- Afegir el control d'accés a l'aplicació. Acabar d'implementar la seguretat en l'aplicació. És una funcionalitat que es va decidir bloquejar durant la fase de desenvolupament però que encara resulta interessant.

- Millorar el sistema de missatges de l'aplicació perquè s'actualitzin en temps real en produir-se. En aquest aspecte s'han estudiat solucions com els websockets.

- Millorar el sistema de missatges de l'aplicació perquè sigui capaç d'enviar correus electrònics diaris amb les dades a tenir en consideració cada dia. D'aquesta manera, el client pot saber, en un sol missatge, què ha de tenir en consideració per al dia o la setmana.

- Implementar el sistema de comentaris sobre els clients que es va descartar inicialment. Tot i que no és una funcionalitat que aportava un benefici gran, és una cosa interessant per al client.

- Millorar el llistat de clients perquè permeti la cerca de clients i perquè permeti més opcions d'ordenació.

- Implementar el control de les vacances dels professors.

- Permetre afegir fotos de retrat dels clients a les seves fitxes.

Conclusions

Un cop finalitzat el projecte, puc afirmar que estic prou content amb el resultat i amb el procés. El procés ha sigut molt enriquidor perquè m'ha permès enfrontar-me amb un projecte real de manera autònoma. He hagut de gestionar el tracte amb el client per a ser capaç de conèixer les seves necessitats i oferir-li solucions adients. He hagut de gestionar els canvis en els requisits per a adaptar-me a allò que és més important i per a solucionar els problemes de la manera que vol el client. Tota aquesta part de la professió és una part que no havia tractat mai i a la que m'he hagut d'adaptar durant el projecte.

En la vesant tècnica estic molt orgullós de la feina feta. He decidit enfrontar-me a tecnologies que coneixia però de manera molt vaga i que volia dominar i conèixer més profundament perquè les considero importants i interessants. Així, he ampliat enormement els meus coneixements d'AngularJS, un framework que té una corba d'aprenentatge molt pronunciada i que és molt ampli. He hagut de llegir codi de les llibreries que he fet servir al projecte, fins al punt d'haver modificat algunes d'elles per a adaptar-les a les meves necessitats. Aquest fet m'ha permès eixamplar els meus coneixements en el framework, però, el que és més important, en com resoldre problemes veient com ho fan altres persones.

També he pogut conèixer un producte nou com és SpringBoot que és la continuació d'un producte que és gairebé un estàndard en el món Java com és Spring. M'ha resultat un producte interessant i molt senzill d'utilitzar, amb una corba d'entrada molt menys pronunciada que AngularJS i que Spring clàsic.

Respecte al resultat del Treball Fi de Grau, tinc una sensació positiva. És cert que un dels requeriments demanats no ha sigut complert però l'objectiu fonamental en començar el projecte era lliurar un producte útil i viable per al client i això s'ha complert. L'aplicació compleix amb els requisits funcionals més importants pel client i resol el problema que va ser origen d'aquest projecte.

Finalment, hi ha un aspecte del desenvolupament del projecte amb el que no estic gaire content. El client va patir problemes burocràtics durant la segona meitat del projecte i no ha trobat temps per fer les proves sobre allò que s'anava lliurant a l'entorn de preproducció per a corregir i reorientar les solucions proposades. Davant aquesta circumstància, les decisions s'han pres, de manera gairebé exclusiva, per l'equip de

desenvolupament. Això és un risc molt gran perquè les correccions i els errors són més costosos quan més tard es detectin. La primera meitat del projecte el client estava molt implicat i això va donar molta empenta i va dirigir el projecte en la direcció correcta i és per això que he trobat molt a faltar aquest interlocutor en la segona meitat del desenvolupament.

Lliurables

Juntament amb aquest document de memòria es lliuren un arxiu zip amb el codi de l'aplicació, i un document d'autoinforme.

L'aplicació està disponible al servidor d'Openshift a l'adreça:

http://pilates-corestudio.rhcloud.com

Aquest és un servidor gratuït i de vegades el servei està suspès si no es reben peticions en un període de temps. Si el sistema es troba suspès, en rebre una petició s'arrenca de manera automàtica, per tant, passats uns minuts l'aplicació torna a estar en funcionament.

El codi també es troba disponible al repositori que he convertit en públic de GitHub a l'adreça:

https://github.com/natete/corestudio

Bibliografia

- Sierra, Kathy, and Bert Bates. *Head First Java*. Sebastopol, CA: O'Reilly, 2005.
- Morrison, Michael. *Head First JavaScript*. Beijing: O'Reilly, 2008.
- McLaughlin, Brett, Gary Pollice, and David West. *Head First Object-oriented Analysis and Design*. Sebastopol, CA: O'Reilly, 2007.
- McLaughlin, Brett, Gary Pollice, and David West. *Head First Object-oriented Analysis and Design*. Sebastopol, CA: O'Reilly, 2007.
- McLaughlin, Brett, Gary Pollice, and David West. *Head First Object-oriented Analysis and Design*. Sebastopol, CA: O'Reilly, 2007.
- Walls, Craig, and Ryan Breidenbach. *Spring in Action*. Greenwich, CT: Manning Publications, 2008.
- Walls, Craig, and Ryan Breidenbach. *Spring in Action*. Greenwich, CT: Manning Publications, 2008.
- Bauer, Christian, Gavin King, and Christian Bauer. *Java Persistence with Hibernate*. Greenwich, CT: Manning, 2007.
- Sierra, Kathy, and Bert Bates. *SCJP Sun Certified Programmer for Java 6 Study Guide (exam 310-065):*. New York: McGraw-Hill, 2008.
- Shaping up with AngularJS. CodeSchool.
- Wikipedia.
- StackOverflow.
- The Complete Bootstrap Masterclass Course. Udemy.
- Learn and understand AngularJS. Udemy.