Disseny del sistema de gestió consular de Portugal

Alejandro Tadeo Romero

Resum— Creació d'una arquitectura base de construcció d'aplicacions web ASP.NET MVC 5 que implementi el control d'autorització, autenticació i suport multi idioma seguint Domain Driven Design. El resultat del projecte ha de permetre iniciar una aplicació zero utilitzant un framework Responsive Html amb un menú administratiu que resolgui la gestió create, read, update, delete (CRUD) de les entitats que han de permetre configurar el model de control d'autorització.

Paraules clau— Metodologies Agils, FrontEnd, Test Drive Developing, Interface, Domain Driven Design, SGBD, Patrons de Disseny, Sistemes de la informació TIC, BackEnd, Framework Entity, CRUD

Abstract— Create a web Applications with ASP.NET MVC 5 that implements the control authorization, authentication and multilanguage support using Domain Driven Design. The objective of the project is to start an application using a framework Responsive HTML with an aministrative menu which resolves the CRUD management of the entities that have to allow setting the model control of authorization.

Index Terms— Agile methodologies, FrontEnd, Test Drive Developing, Interface, Domain Driven Design, SGBD, Design Patterns, Information System TIC, BackEnd, Framework Entity,CRUD

1 Introducció

DES DELS anys 80 fins avui dia, la tecnologia ha tingut un paper rellevant en la transició dels paradigmes de gestió de la informació i d'equips. Les organitzacions s'han vist obligades a implantar nous sistemes de treball més corporatius, que permeten obtenir una bona administració i comprensió del sistema d'informació que ofereixen millores en la coordinació de la informació.

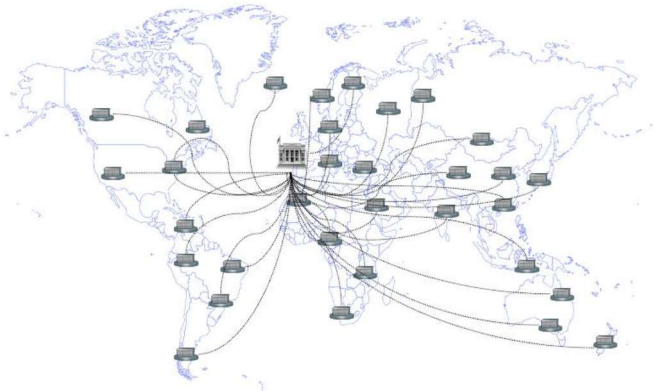
Aquest projecte pretén desenvolupar una infraestructura basada en una arquitectura de construcció d'aplicacions web mitjançant la tecnologia model, vista i controlador (MVC). Posteriorment, servirà de base d'un projecte de més alt nivell que consisteix en l'actualització de la gestió de la informació (TIC) a tota la xarxa consular de Portugal. Esmentar, que al tractar-se d'un projecte internacional cooperatiu entre dos equips distribuïts, sorgeixen complexitats que s'hauran de resoldre aplicant nous conceptes teòrics i metodologies innovadores per tal d'aconseguir obtenir èxit en la implantació del nou sistema a les oficines consulars anomenat eSGC, amb l'objectiu de donar una resposta a les necessitats operacionals.

A continuació, l'organització del document contempla des de la introducció del projecte fins a plans de millora, contemplant: plantejament, requisits funcionals, test i qualitat, punts de discussió i millores que mostren la presa de decisions al llarg del projecte, tecnologia utilitzada, funcionalitats, valoracions, resultats estimats i aconseguits.

1.1 Antecedents

La xarxa consular comprèn 118 oficines arreu del món, on actualment hi ha dues versions de l'aplicació sistemes de gestió consular (SGC) operatives. Una versió VB amb SGBD Oracle 9i, que opera en 69 oficines consolars i una versió web desplegada sobre servidors locals amb SGBD SQLServer veure a l'annex (1).

Atès que no hi ha comunicació entre totes les bases de dades (la imatge 1 és il·lustrativa per centrar la idea), la finalitat és proveir d'un sistema que sigui capaç d'unificar tots els consolats, així com verificar a Ministérios dos Negócios Estrangeiros (MNE) les inscripcions duplicades dels usuaris a causa de ser un sistema distribuït.



1.Xarxa Consular (il·lustratiu)

- *Alejandro.Tadeo@geyce.es*
- *Menció Enginyeria del Software.*
- *Tutors:*
  - *Joan Vilaseca (Director Geyce Biometrics)*
  - *Josep M.Basart (Enginyeria de la Informació i de les Comunicacions).*
- *Curs 2015/16*

2 PROPÒSIT DEL TFG

L'objectiu principal és proporcionar una aplicació web per a gestionar entitats i permetre configurar els models de control d'autorització i autenticació, oferir una versió base que serà implementada pel sistema eSGC, i obtenir una millor flexibilitat en la persistència i actualització de la informació i unificar tota la informació a la central a Lisboa.

3 ABAST DEL PROJECTE

Aconseguir una arquitectura d'aplicacions web utilitzant ASP.NET MVC 5 que implementi el control d'autorització i autenticació i suport multi idioma seguint la filosofia DDD. El resultat del projecte ha de permetre iniciar una aplicació zero utilitzant un framework responsive Html amb un menú administratiu que resolgui la gestió CRUD de les diferents entitats, configurant el model de control d'autorització.

3.1 Objectius del projecte

L'aplicació zero ha de ser implementada basada a complir les següents directrius:

- L'aplicació ha de gestionar el model de configuració d'autenticació i autorització i verificar els processos d'accés amb la utilització de software OpenSource.
- L'aplicació ha de gestionar un patró CRUD sobre l'arquitectura amb les entitats necessàries per administrar usuaris, oficines, rols, serveis i permisos.
- Arquitectura Domain Driven Design (DDD) sobre ASP.NET MVC5.
- Realitzar components de proves de regressió amb Test Driven Development (TDD).

3.2 Proposta de solució

Per tal d'aconseguir el propòsit caldrà una correcta persistència de la informació i desnormalització. Obtenint una independència entre la capa lògica (FrontEnd) i la part física (Backend) del sistema mitjançant l'aplicació DDD que millorarà la flexibilitat de gestió, veure l'annex (2).

3.3 Enfocament al projecte TFG

Aquest projecte s'integra com a iteració zero d'un projecte de desenvolupament del sistema de gestió consular pel govern portuguès anomenat eSGC que l'empresa GEYCE en consorci amb l'empresa GMV skysoft ha de construir pel ministeri d'afers exteriors portuguès (MNE).

Pel desenvolupament se segueix TDD (test-driven development) amb la plataforma de desenvolupament integrada Microsoft Azure i IDE Visual Studio. Implementat seguint una metodologia iterativa integrada en Foundation Team Services amb un repositori GIT.

3.4 Estat de l'art

A continuació s'explicarà què és l'arquitectura DDD i perquè formarà l'esquelet principal del projecte [3].

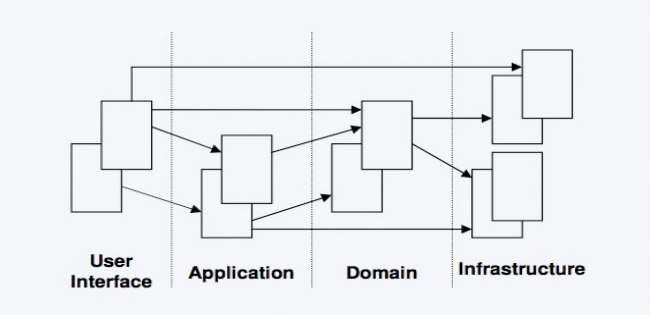
Eric Evans, autor del llibre Domain-driven design[1], explica un concepte en l'àmbit teòric que pretén extrapolar la dificultat que moltes empreses troben quan volem gestionar rols, permisos, serveis, dades, informació, tràfic de manera complexa i mitjançant una interfície. Ens planteja com a alternativa viable un nou concepte arquitectònic i també, una manera de treballar la qual anomena Domain-Driven Design. En endavant DDD , no només és un estil arquitectural, sinó una forma de treballar i d'afrontar el projecte a un nivell de treball enfocat al treball en equip al llarg del desenvolupament.

Com a conseqüència, cal crear un Llengüatge Ubic a utilitzar amb els Experts del negoci. No obstant això, DDD també implementa una sèrie de patrons de disseny i estils d'arquitectura específics, per tal de facilitar aquesta proposta i ser dut a terme mitjançant un programari modular de gestió web, s'ha de realitzar un focus al cor de l'arquitectura anomentat domini (domain), ja que els altres elements, complements i comportaments giren al voltant d'aquest nucli tal com mostra la imatge 2.

A continuació, el patrons de l'arquitectura DDD:

- Repository.
- Entity.
- Aggregate.
- Value-Object.
- Unit-Of-Work.
- Services.

També incorpora un nou sistema d'arquitectura que treballarà al costat d'aquests patrons de disseny per tal d'aconseguir un desacoblament ideal al voltant del domini que ell anomena Arquitectura N-Capes, formada per les capes vistes i interfícies, aplicacions, serveis, domini i infraestructura, i que utilitzant objectes DTO es comunicaran entre elles.
Aquest Data Tranfer Object, en endavant DTO són objectes simples que no han de contenir lògica de negoci i la seva finalitat és transportar dades entre els diferents processos a les diferents capes. La motivació de la seva utilit-



2. Arquitectura de Capes DDD

zació té relació amb el fet que la comunicació entre processos és usualment realitzada mitjançant interfícies remotes, més conegudes com a serveis Web, on cada crida és una operació molt costosa en recursos. Com la major part del cost està relacionada amb el temps d'enviar un paquet de dades (round-trip) entre el client i el servidor, una forma de reduir el nombre de trucades és utilitzant objectes DTO que afegeixen les dades que s'han de transferir de cada crida, però que són lliurats en una sola invocació.

Tot això cobra sentit a causa del cor de l'aplicació, el model de domini que com és una projecció directa d'aquest llenguatge de gestió de negoci, permet agilitzar la gestió i el seu manteniment perquè troba ràpidament àrees incorrectes en l'entorn que l'envolta, com Eric Evans diu, si coneixem el nucli, el seu entorn és merament un embolcall que haurem d'embolicar com si fos un regal [2].

3.5 Discussió i anàlisis DDD

Fins ara, qualsevol persona amb coneixements en l'ambit de l'entorn web podria considerar la possibilitat de realitzar aquest mateix projecte utilitzant Web Page amb una base CRUD. Existeixen dues grans raons per les quals s'ha de considerar l'utilització de l'esquelet DDD i no simplement una filosofia basada només en CRUD.

La primera consideració es que DDD ajuda a gestionar eficientment el comportament complex del sistema, però què el fa realment interessant? Ajuda a treballar a través de les regles de negoci i comportaments especials, per exemple, el tractament d'un client, podria obtenir dades d'aquest client per determinar si validar l'accés i omplir la taula amb la referència de la transacció, que a diferència del sistema CRUD al ser orientat a dades i no conèixer les regles de negoci, no seria possible a causa de la transacció que es realitza a l'interfase.

La clau del DDD és que inclou la lògica del negoci des del domini com a mètodes dins de les classes d'entitat, aprofitant la programació orientada i la restricció de comunicació entre les diferents capes, és a dir, el tractament de la informació mitjançant la utilització de DTO, serà coneguda només per les capes inferiors i com si fos una caixa negra. A més, els objectes en lloc d'implementar comportaments de negoci dins dels objectes, aquests no contenen cap estat, a diferència del CRUD.

La segona raó a considerar l'utilització DDD, a diferència d'un model basat en dades és que permet treballar la lògica sense tenir present la persistència donant certa flexibilitat que fa guanyar temps als projectes, és a dir, no cal esperar a les anàlisis prèvies de bases de dades perquè l'arquitectura DDD permet independentment de la persistència acoblar la solució feta. Això va ser un punt molt important a tenir present i que va decantar la balança a la utilització de Data First en comptes de Code First quan es va escollir l'Entity Framework a utilitzar.

Finalment, cal destacar que DDD no és una metodologia ni una tecnologia, sinó filosofia que estructura el treball i exigeix prendre desicions de disseny per aconseguir major acceleració i flexibilitat en el management de transaccions i gestions complexes en projectes software. Així com, traça el camí per la integració entre tècnics del domini i experts del negoci proposant la creació d'un llenguatge Ubic. Punt molt important a causa de que actualment no poden existir projectes complexos sense contenir barreres idiomàtiques.

Per tant, DDD facilita poder treballar de manera paral·lela i permet tenir una major flexibilitat en el desenvolupament [3].

3.6 Avantatges i desavantatges del DDD

Un possible desavantatge que podria tenir utilitzar aquesta arquitectura és que sol ser restrictiva a l'hora de portar-la a terme en la pràctica, en ser un sistema complex, embarcar-se en la seva programació pot arribar a ser més un problema que una solució si no respecta les directrius preestrablertes pel DDD. A més, cal ser conscient que exigeix un coneixement avançat de programació i sobre el seu desenvolupament teòric, així com una correcta sincronia i entesa entre els diferents equips.

És recomanable la utilització de DDD només per projectes amb una certa complexitat del model i gestió arquitectural és més gran que en una aplicació orientada a dades. Degut a que DDD implica mantenir un model de capes amb gran quantitat d'aïllaments i encapsulacions, es genera una gran quantitat de codi i pot arribar a tenir un cost relativament alt.

Però tenint present aquests desavantatges, sense cap dubte DDD ofereix grans avantatges com són [3]:

La comunicació: Totes les parts de l'equip poden fer servir el model de domini i les entitats que defineix per comunicar coneixement del negoci i requeriments, fent ús d'un llenguatge comú.

Extensibilitat: En utilitzar N-capes desacoblades el domini està completament desacoblat de les capes d'infraestructura i vista, sent més fàcil així escalar i evolucionar la tecnologia del projecte.

Testing: DDD facilita el testing, degut a que la tendència de disseny és a desacoblar els objectes de les diferents capes de l'arquitectura permeten realitzar tests unitaris més concrets.

Flexibilitat: Alineament amb el domini, ja que reutilitza serveis comuns amb interfícies estàndard augmentant les oportunitats tecnològiques i de negoci reduint costos.

Acoplament: Abstracció ja que els serveis són autònoms i accedeixen a través d'un contracte formal que proporciona baix acoblament i abstracció mitjançant les capes inferiors.

Intuïtiva: Els serveis exposen descripcions que permeten a altres aplicacions trobar i determinar la seva interfície automàticament.

Manteniment: En realitzar la gestió mitjançant el desacoblament entre component, capes i nivells permet mantenir amb facilitat un codi net i pulit. També facilita la correcció d'errors (bugs) degut a que la lògica del negoci és només en el domini.

4 METODOLOGIA

La metodologia àgil utilitzada al llarg del projecte, serà basada en el procés àgil conegut com a SCRUM [10]:

L'elecció d'aquesta metodologia àgil és millorar la comunicació entre els dos equips Geyce i GMV, i també aconseguir incrementar l'eficàcia i eficiència a l'hora de treballar en el desenvolupament del projecte. Tant GEYCE com GMV necessiten obtenir resultats predictibles ràpids l'un de l'altre a causa del distanciament i la barrera idiomàtica.

El projecte s'ha planificat dintre del termini de sis mesos de treball com mostra l'Annex A2, per tant, sis blocs de vint-i-dos dies laborals amb estimació d'iteracions mensuals des de 4 a 5 setmanes.

A més, juntament SCRUM, s'utilitzarà la metodologia de treball anomenada TDD [4]. Un procés basat en la idea de realitzar proves unitàries al codi que s'ha de construir, a diferència del procediment que es fa servir habitualment (construir el codi i després realitzar les proves). TDD estableix que primer cal fer una prova i tot seguit desenvolupar el codi que la resol, així tots dos equips podran assegurar-se que tot el codi adquirit de l'altre equip és fiable i correcte perquè durant el seu desenvolupament ha sigut prèviament testejat i realitza la funcionalitat que ha de fer.

És important destacar que s'ha realitzat un gran esforç entre tots els integrants del grup i equips per tal d'entendre i arribar a un llenguatge Ubic Anglès que es va acceptar al llarg de tot el projecte i així treballar de manera cohesiva.

4.1 Planificació del projecte

Aquest apartat dóna una visió general de l'organització al llarg de les diferents fases del projecte (veure annex 3), a més de les eines utilitzades per desenvolupar el projecte com; hardware, metodologies, software, requisits i recursos.

Anàlisis i especificacions: En aquesta fase es van extreure requeriments, realitzar el modelatge del sistema, establir bases de treball i estudiar alternatives arquitectòniques dintre de la solució DDD i aconseguir una proposta framework responsive per ser utilitzada com interfície d'usuari i que pugui ser compatible amb l'entorn tecnològic a utilitzar ASP.NET.

Gestió primera entita: Va caldre refinar el diagrama de classes de l'estructura de l'aplicació, veure Annex 4. Com desenvolupar gran part de l'arquitectura de l'esquelet N-capes i implemantar una primera entitat Users en l'estructura DDD mitjançant una interfície login, la finalitat va ser comprovar que la infraestructura actuava correctament així com l'aplicació.

Completar resta entitats: Aquesta etapa va assolir l'entrada de la resta d'entitats per aconseguir obtenir el model d'autenticació i autorització a gestionar com; administrador, rols, permisos, oficines i serveis, així com realitzar una primera capa de servei tal que permeti l'autorització i autenticació.

Finalitzar codificació del model: Finalitzar la codificació i conflictes no previstos al llarg del desenvolupament del projecte, en finalitzar-la mostar una versió beta d'aplicació zero.

Desplegament d'aplicació: Desplegament de l'aplicació a la plataforma Azure per acabar de refinar el correcte funcionament mitjançant tests funcionals. En finalitzar-los l'aplicació serà enllestida per formar part d'una iteració zero del projecte de desenvolupament eSGC.

Fins al final, tots els resultats van ser satisfactoris, tot i que al llarg del desenvolupament van sorgir conflictes, aquests es van resoldre sense cap mena de repercussió en la planificació del projecte.

4.2 Tecnologies utilitzades

El projecte s'aborda amb les següents arquitectures i tecnologies:

- Windows Server 2012 R2.
- SGBDR Microsoft SQLServer 2014.
- Microsoft .NET Framework 4.5.
- ASP.NET MVC 5 amb AJAX.
- Codi Programació C# amb EntityFramework.
- Microsoft SQL Server Reporting Services 2014.
- Front-end Framework Bootstrap amb Inspinia.
- SourceTree Atlassian.
- Version Control GIT.
- Plataforma Cloud Azure.
- Microsoft Team Services.
- Enterprise Architect.

4.3 Requisits funcionals

Per tal que l'aplicació doni una resposta acurada a les necessitats reals de l'aplicació i al client. Cal considerar els següents requisits relacionats amb el contingut, estructura i usabilitat de l'aplicació web.

R.F1: L'aplicatiu web ha de regular l'accés d'usuaris basat en permisos que puguin ser directament associats amb perfils o grups als quals pertanyen.

R.F2: Els permisos han de poder ser configurats per reflectir les necessitats del model de funcionament i processos de negoci suportats.

R.F3: Ha de permetre delegar responsabilitats operatives d'un usuari a un altre. Aquesta atribució de responsabilitats ha de ser capaç de revertir-se.

R.F4: El sistema ha d'enviar dades mitjançant un format standaritzat PDF/Excel.

RF5: El sistema ha de tenir les funcions de suport necessàries per a la gestió de les taules.

RF6: El sistema ha d'incloure mecanismes per extreure un registre històric i eines per visualitzar-les a l'entorn web.

RF7: El sistema no ha de permetre el registre de sol·licituds d'actuacions consulars als usuaris que no estan registrats.

RF8: La clau d'inscripció ha de ser única.

RF9: El sistema ha de ser capaç d'agregar les dades a diversos nivells.

RF10: El sistema ha de permetre la restricció visual en funció del rol i privilegis que conté l'usuari.

RF11: El sistema ha de gestionar l'entitat sota una filosofia CRUD.

RF12: L'aplicació ha de ser compatible amb explorador Firefox, Chrome i Explorer 7 i posteriors versions

RF13: L'aplicació web ha de contenir filtres de selecció.

4.3.1 Requisit no funcionals

R.NF 14: L'aplicació ha de ser easytouse

R.NF 15: Les Views han de ser sencilles i intuïtives.

R.NF 16: L'aplicació ha de mantenir les dades emmagatzemades de manera segura i fiable.

R.NF 17 : El desenvolupament de l'aplicació web s'ha de realitzarà en ASP .NET.

4.3.2 Requisit hardware

RH.18: L'aplicació ha de ser compatible amb les instal·lacions als consolats portuguesos.

RH. 19: L'aplicació ha de ser capaç de treballar amb temps de resposta ràpids amb capacitat d'Internet des de 512kbs.

RH.20: L'aplicació ha de poder instalar-se en un disc 20 GB.

4.3.3 Requisit software

R.S 21: L'aplicació ha de ser compatible amb els sistemes operatius; Windows XP en endavant.

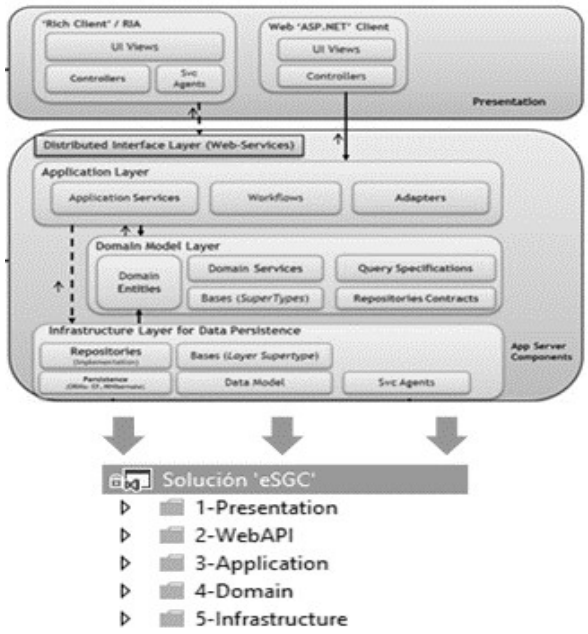
R.S 22: El software per desenvolupar el projecte ha de prioritzar OpenSource.

4.4.Desenvolupament i implementació

A continuació, aquest apartat explica com s'ha desenvolupat i aplicat a la teoria d'Eric Evans a la pràctica.

4.5 Integració

En aquesta etapa es va crear l'estructura bàsica DDD d'Eric Evans, proposada en l'àmbit teòric [8], en la que es



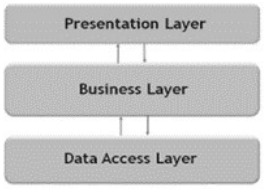
3. Aquitectura DDD

diferencien els components de cada capa per tal d'aconseguir una cohesió, i aproximar un nivell d'abstracció similar en totes les capes, veure imatge(3) .

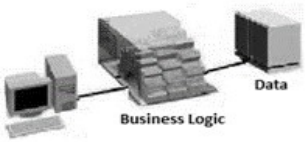
La imatge mostra l'estructura creada simulant les diferents capes mitjançant la creació de diferents carpetes. Mitjançant les carpetes i 21 projectes es gestiona totes les dependències del projecte i transaccions, reduint la complexitat en la comunicació entre els components dins la seva pròpia capa. I amb la utilització d'interfícies es realitza la comunicació entre els diferents nivells de l'aplicació tal com mostra l'imatge.

4.6 Layers i tiers

Existeix una gran diferència entre dos conceptes en l'arquitectura DDD, i és que són capes i nivells; els quals s'expliquen de manera breu a continuació, amb la finalitat d'evitar confusions, quan es parla de les diferents capes DDD i nivells; al llarg del projecte, veure imatge (4).



Layers



N-Tier

4. Aclariment conceptes lògic i físic.

Capes (Layers): Encarregades de la divisió lògica de components i funcionalitats, independentment de la localització física dels components. Implica una separació lògica.

Nivells (Tiers): Encarregats de la distribució física de components i funcionalitats en diferents servidors. Implica una separació física N-Tier.

4.7 Capa domini

La capa Domini (veure imatge 5) és el cor del software i nucli del DDD. Aquesta capa implementa la funcionalitat principal del sistema i encapsula tota la lògica de negoci rellevant, dins dels seus mètodes. La raó d'implementar aquesta lògica és aconseguir separar de manera clara els comportaments de les regles del negoci dels detalls d'implementació d'infraestructura; d'aquesta manera aconseguim aïllar el domini de l'aplicació.

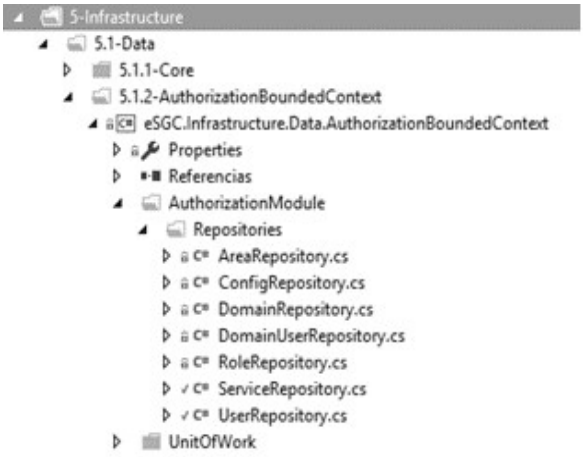
5.Capa Domini

Cada Entitat tindrà una carpeta, que a la seva vegada, estarà formada per un mínim de 4 classes:

- **IClase.cs –** Interfície: Per accedir a la classe Root de l'Objecte a la capa d'Infraestructura.
- **Clase.cs –** Definició: Declaració de l'entitat i les sever relacions.
- **ClaseOrders.cs –** Intercomunicador: Conté les relacions d'ordenació entre les capes.
- **ClaseFactory.cs –** Constructor: Mètode per crear l'objecte i definir a la interfície.
- **ClaseSpecification.cs –** Intercomunicador: Conté les relacions de comunicación entre les capes.

4.8 Capa infraestructura

La imatge (6) a continuació mostra l'estructura de la capa d'infraestructura.

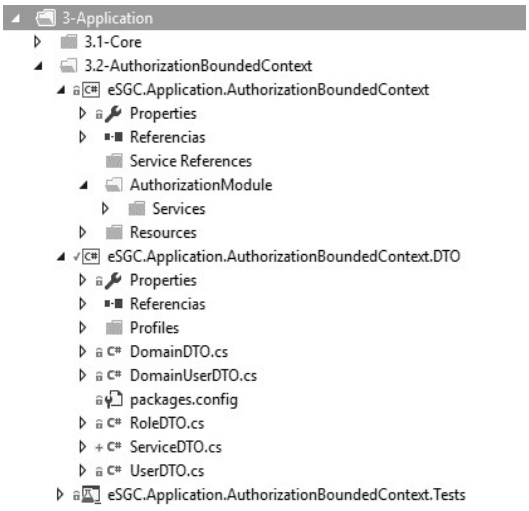


6.Capa d'Infraestructura

A causa de l' importància que té l'accés a dades en l'aplicació, la capa d'infraestructura és l'encarregada d'encapsular la lògica de persistència de dades, requerida per accedir a les fonts de dades per l'aplicació. Centralitzant la funcionalitat d'accés amb un desacoblament entre la tecnologia, utilitzant UoW(Uni of Work). Així, contindrà les classes de tipus repositori recolzades sobre mapeig de dades ORM (Object-Relational mapping).

4.9 Capa d'aplicació

Aquesta capa d'Aplicació (veure imatge 7) no inclou cap lògica de negoci, només coordinació relativa a requeriments tècnics de l'aplicació. Principalment, tot i que aquesta capa és molt senzilla, té un impacte considerat; ja que defineix les tasques que suposadament, ha de realitzar el software. Per exemple; fer crides a la capa domini, i infraestructura per executar tasques del domini.



7.Capa d'Aplicació

No obstant això, hi ha tasques que són exclusives de la mateixa capa d'aplicació com conversions de dades, implementació mitjançant els objectes DTO mitjançant dues subcapes.

Subcapa AuthorizationBoundedContext: Encarregada de la coordinació entre la capa domini i infraestructura mitjançant el patró Service.

Subcapa AuthorizationBoundedContext.DTO: Encarregada de la coordinació amb la capa de presentació i tractament d'objectes; i formada per les dades cap a DTO, mitjançant el patró adapter i un mètode de Entity Framework.

4.10 Capa web

Aquesta capa (imatge 8) presenta a l'usuari els conceptes de negoci mitjançant una interfície d'usuari, facilitant l'ús dels processos, dóna la informació de la situació de processos de negoci i implementació de les regles de validació, així com la interacció amb l'usuari mitjançant l'utilització de pagines Web amb arquitectura MVC5.

8.Capa Web

Fins ara, només s'ha definit l'esquelet bàsic del projecte amb les classes estructurades, organitzades i buides. El propòsit rau en utilitzar els tests amb TDD, i anar a poc a poc omplint totes les funcionalitats del projecte amb els mètodes de gestió CRUD. Llavors, cal començar a codificar el pla de proves test, i controlar mitjançant diferents tipus d'Assert, la correctivitat dels tests, i la seva veracitat, obtenint el valor esperat. En finalitzar el test s'obtenen les diferents classes amb mètodes i funcionalitats idònies, amb la seguretat de que es fa el que s'ha de fer, de manera correcta i assegurant bon resultat.

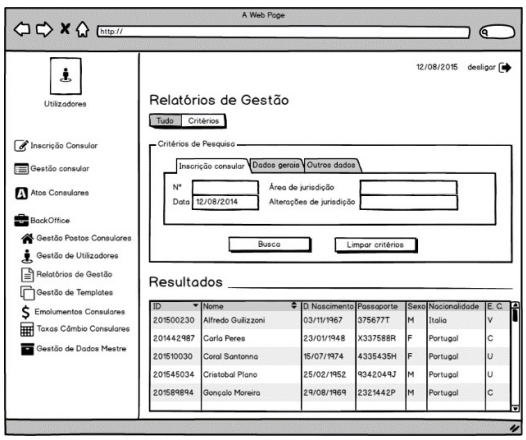
5. DISSENY DE LA WEB

Per dissenyar l'aplicació es van plantejar diferents prototips ViewModels dels que es va fer un procés d'acceptació. Els dissenys aprovats (veure imatge 9 i 10) inclouen dues parts principals de l'aplicació que són; un menú de login inicial per entrar a l'aplicació i el menú principal d'inici un cop autentificat i validat.

El primer mockup és un menú principal que conté d' un login mitjançant dos camps de text que l'usuari utilitzarà per verificar la seva identitat a l'aplicació. Com exemplifica l'annex 4.

9. Login

Aquest segon mockup conté un menú lateral a l'esquerra que conté les diferents opcions d'àrees, un contenidor a la dreta del menú on es carrega la informació que l'usuari vol consultar mitjançant gràfiques, combos o imatges, una capçalera amb diversos enllaços d'interès i un peu de pàgina amb informació del copyright del projecte.



10. Mockup Aplicacio web

6. TDD I TEST

Al llarg del projecte, tot el desenvolupament es desenvolupa mitjançant TDD. Un cop realitzat el codi utilitzant TDD es var realitzar les proves de diversos tipus per assegurar la qualitat del producte i solucionar possibles errors del codi.

6.1 Proves de regressió

El propòsit de les proves és demostrar el compliment dels requisits dels sistemes. Aquestes proves duran a terme al llarg del projecte eSGC.

Proves funcionals: Aquestes proves estan dissenyades per demostrar el funcionament correcte de l'aplicació web i són realitzades a la plataforma de preproducció Azure. Van consistir a realitzar diferents transaccions a l'entorn web com si fos un usuari real.

Proves unitàries: Comprovar el correcte funcionament d'un mòdul de codi. Per assegurar que cada un d'ells funcioni correctament per separat amb el tractament de dades. Aquests es van realitzar a cada capa.

Proves d'integració: Assegurar el correcte funcionament del sistema o subsistema en conjunt. Aquestes proves es van realitzar a la capa d'aplicació per assegurar el funcionament de tota l'estructura.

Proves d'acceptació: La implementació actual de les proves d'acceptació està a càrrec del Ministeri d'Afers Exteriors.

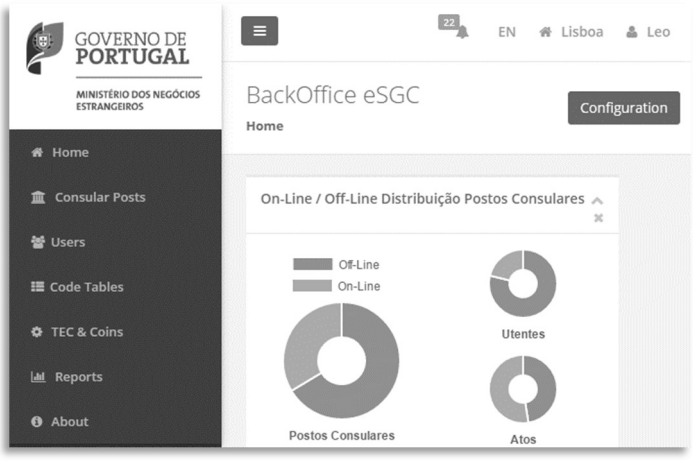
7. RESULTAT

A continuació, en les següents seccions es mostra les funcionalitat dels resultats obtinguts.

Menu Login: Aquest menú verifica l'autenticació i validació de l'usuari com es va plantejar als casos d'ús, veure l'annex 4.

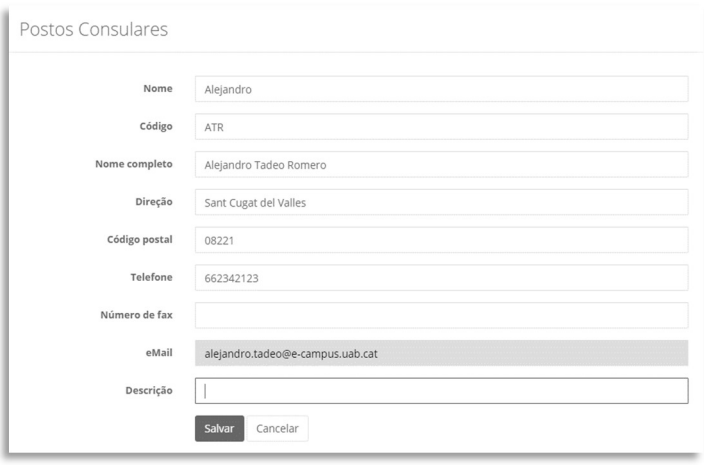
11.Login Autenticació eSGC

Les següents imatges mostren alguns dels resultats; menú inicial, alta, modificar, esborrar i editar, veure annex 5



12.Menú Inicial BackOffice.

13.Gestió d'Usuari



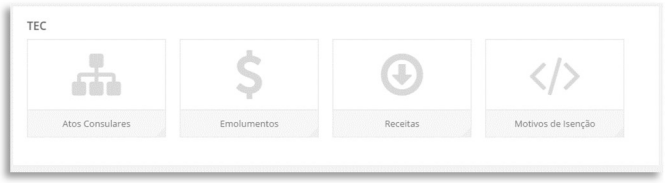
14.Crear Nou Usuari

A continuació, algunes utilitats requerides explicitament pels requisits de l'usuari; filtres, ordenació, combos, links, export .PDF/Excel, multilanguatge, oficines i busqueda.

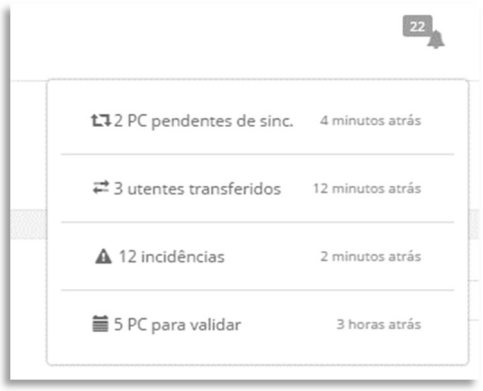
15.Àrea i Multidioma

16. Busqueda i Exportar PDF/ Imprimir

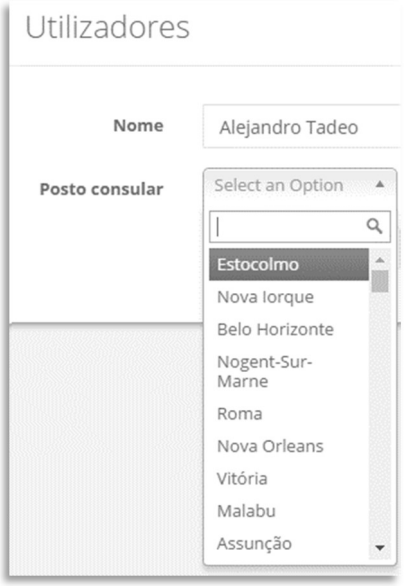
17. Enllaços d'accés



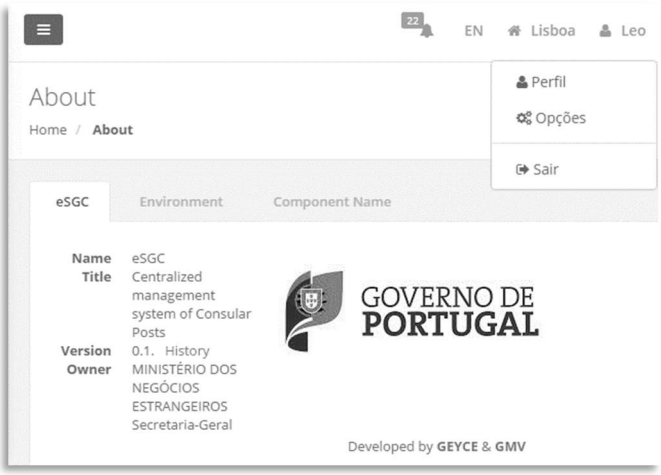
18. Navegació EasyToUse



19.Alertes i Notificacions



20. Combo selecció



21.Multilanguatge i Gestió Perfil

8.SEGURETAT APLICACIÓ

La seguretat de l'aplicació és gestionada mitjançant el propi frameworks .NET. Només cal configurar-la en funció de la necessitat: directiva de seguretat, administrant certificats, signar arxius digitals, verificació, firma. Totes aquestes eines s'instal·len automàticament amb VisualStudio. Només cal utilitzar el símbol de sistema.

D'altra banda, s'ha realitzat dues validacions; una a la capa aplicació i l'altra a la base de dades, d'aquesta manera poder assegurar la persistència de dades.

També s'utilitzen usings per tal de controlar la gestió de la base de dades, és a dir, la instrucció using garanteix que la base de dades es tanca en cas d'error, encara que es produeixi una excepció mentre es crida els mètodes de l'objecte.

9 CONCLUSIONS

El projecte està finalitzat i l'aplicació zero implementa N-capes DDD és, ja, el punt de sortida dels 6 sprints posteriors del eSGC, s'ha obtingut una aplicació web ASP.NET que implementa una gestió d'autorització, autenticació multi idioma seguint Domain–Driven Desing, desenvolupada mitjançant una tecnologia web MVC i construint tot l'esquelet amb TDD.

Cal destacar que des d'un inici, vaig tenir la sensació que DDD era enrevessat d'entendre i implementar a un projecte real complexe, però un cop donat el primer pas amb la lectura dels capítols del llibre d'Eric Evans, així com el llibre de César de la Torre, i el suport d'un equip com a GEYCE, cada cop estic més convençut que és una bona solució per desenvolupar un projecte tant complex com és dissenyar un sistema de gestió consular (eSGC) perquè implementar DDD implica la necessitat d'utilitzar el seny constantment. Es tracta d'un enfocament de la creació de software personalitzat i propi. Això, en fases preliminars genera confusió, però mica en mica facilita les funcionalitats més complexes en simplificar la feina, reduint tant en l'àmbit de codi la interacció i dependència, com entendre's entre els diferents equips.

Normalment, l'inici d'un projecte mai és exempt de problemes. A més, s'ha d'assumir (sobretot en la planificació inicial) que al projecte tot inici conté conflictes i problemes, perquè inicialment és un enfocament pragmàtic per al desenvolupament de software, però amb temps i paciència redueix l'acumulació de conflictes:

Durant el desenvolupament d'aquest TFG, he comprovat que acadèmicament, se'ns prepara per poder adquirir els coneixements necessaris per poder afrontar l'entorn laboral un cop finalitzada la universitat. A més, el TFG obligar posar en pràctica els coneixements apresos a la carrera. Tot canvia quan et trobes treballant en una empresa perquè t'adones que la teoria apresa no sempre va en consonància amb la realitat empresarial, és a dir, no només s'aprèn durant la carrera on ens preparem seguint conceptes per comprendre a les organitzacions, a tenir empatia amb els usuaris i ser assertius per arribar a ser bons enginyers, però no t'adones del que suposen tots aquests conceptes fins que acabés treballant en alguna empresa.

Esmentar, que, aquest període a GEYCE, he sigut partícip en una nova etapa que ha embarcat l'empresa, apostant per les noves tecnologies i metodologies àgils, és a dir, he viscut tota la transició des del naixement d'un projecte; anàlisis de tecnologies, recerca d'informació, aprenentatge de metodologies de treball i bones pràctiques i assumir responsabilitats dintre d'un projecte real com eSGC.

Algú va dir: "Cap camí fàcil et portarà a alguna cosa que valgui la pena – anònim" i un bon exemple és GEYCE que al llarg de tot el projecte TFG ha implantat noves tecnologies com Domain-Driven Development i metodologies àgils sobre Team Service en un projecte tan complex com és eSGC . Realitzant una aposta de futur pel treball ben fet que va més enllà del nivell econòmic, creant llaços entre diferents equips de treball amb barreres idiomàtiques i tenint en consideració qualsevol suggeriment ben fonamentat. Geyce sens dubte és un model a seguir.

Per finalitzar, un possible pla de futur per aquest projecte inclouria una nova funcionalitat: implementar un sistema impersonate, que facilitaria iniciar a un usuari a l'aplicació però sent la identitat real d'un altre, agilitzant la resolució de possibles conflictes remotament. Caldria desenvolupar el tractament a les diferents n-capes de l'aplicació i activar tres funcions API necessàries per realitzar el tractament d'usuari com LogonUser per iniciar l'usuari a l'equip i autenticar-lo, DuplicateToken que conté la informació de seguretat i protocols de l'usuari al sistema i RevertToSelf per terminar la suplantació a l'aplicació vigent i gravar un registre a la base de dades.

AGRAÏMENTS

Estic molt agraït a tot l'equip GEYCE, per tota l'ajuda, consells i orientació al llarg d'aquest projecte, que sens dubte, no seria possible sense ells. Especialment a Joan Vilaseca i Josep M.Basart per ensenyar-me que la perseverança és la virtut per la qual les altres virtuts donen el seu fruit.

Finalment, agrair a Lidia Garcia tot el seu suport, confiança, i amor durant tots aquests anys, sens dubte, res hauria sigut possible sense ella. Gràcies.

BIBLIOGRAFIA

[1] Domain-Driven Design Reference (2011) Definició i patrons, Consultat el 12 de Febrer de 2016: https://domainlanguage.com/ddd/patterns/DDD_Refrence_2011-01-31.pdf

[2] Domain-Driven Design Eric Evans (2013) Exemple web DDD. Consultat el 13 de Febrer de 2016: http://dddcommunity.org/fredriksson_rayner_evans_2013.

[3] Domain-Driven Design Quickly Eric Evans (2006) Floyd Marc

[4] Guia de arquitectura DDD César de la Torre (2010) Imatges i conceptes. Consultat el 1 de Març de2016: http://download.Guia_Arquitectura.pdf

[5] Ministerio dos negocios estrangeiros (2015) Documentació. eSGC_ MNE-2015-eSGC_3_Cadnc_ANEXO A-CaractSTecns_15.

[6] Proposta Fornecimiento de um sistema integrado de funcionamiento (2015) Documentació projecte.

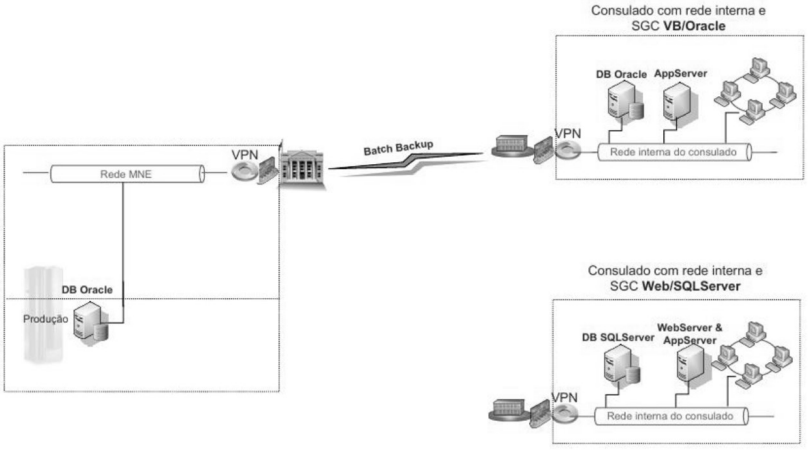
[7] Apresentaçaö e anàlisi das propostes(2015) Imatges i Conceptes. eSGC_CPlpqai_MNE-2015-SGC_1_.03.02_vDEF.

[8] SCRUM (2013) Documentació. Consultat el 2 de Febrer de 2016. http://proyectosagiles.org/que-es-scrum/

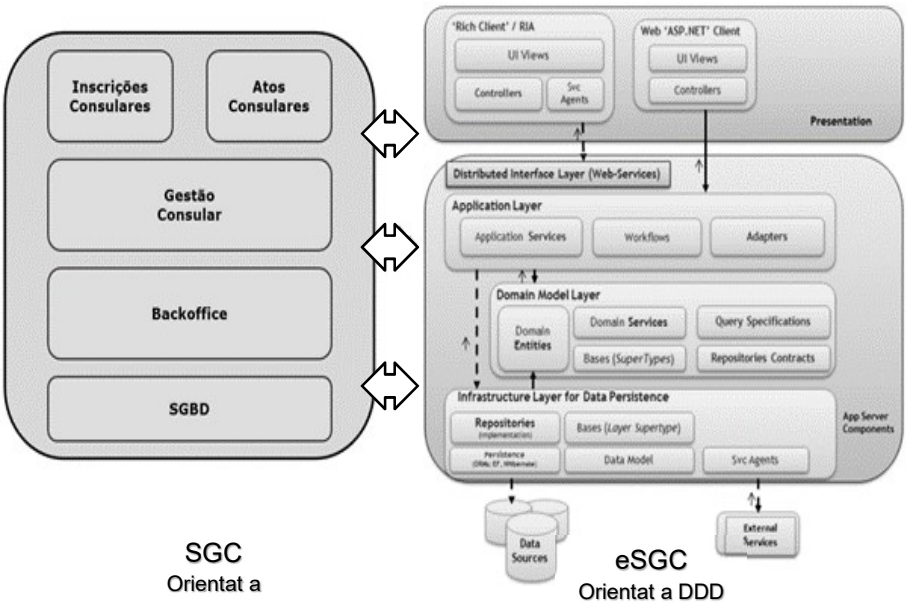
[9] Introduccion TDD (2011) Consulta 24 de Febrer de 2016. http://blog.lordudun.es/2011/04/introduccion-a-tdd-test-driven-development/

[10] Ninject.MVC5 (2014) NuGet. Consultat el 4 de Maig de 2016: https://docs.nuget.org/

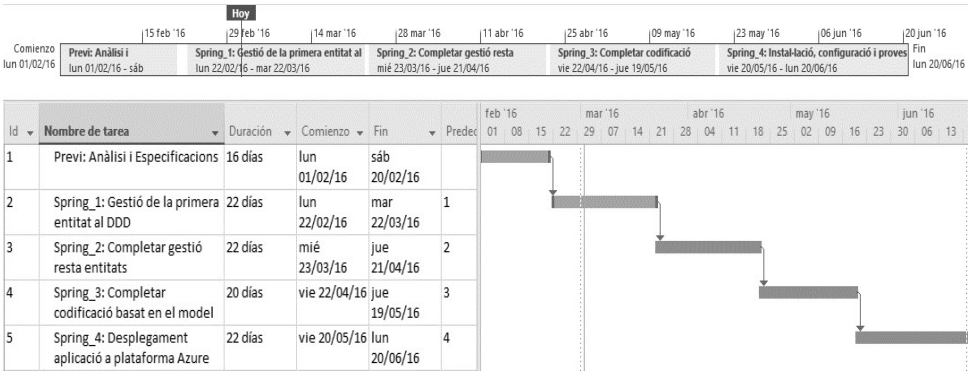
ANNEX 1. ESQUEMA ACTUAL DEL SISTEMA DE GESTIÓ CONSULAR.



ANNEX 2. PROPOSTA ARQUITECTURA SOLUCIÓ – DESIGN-DRIVEN DEVELOPMENT

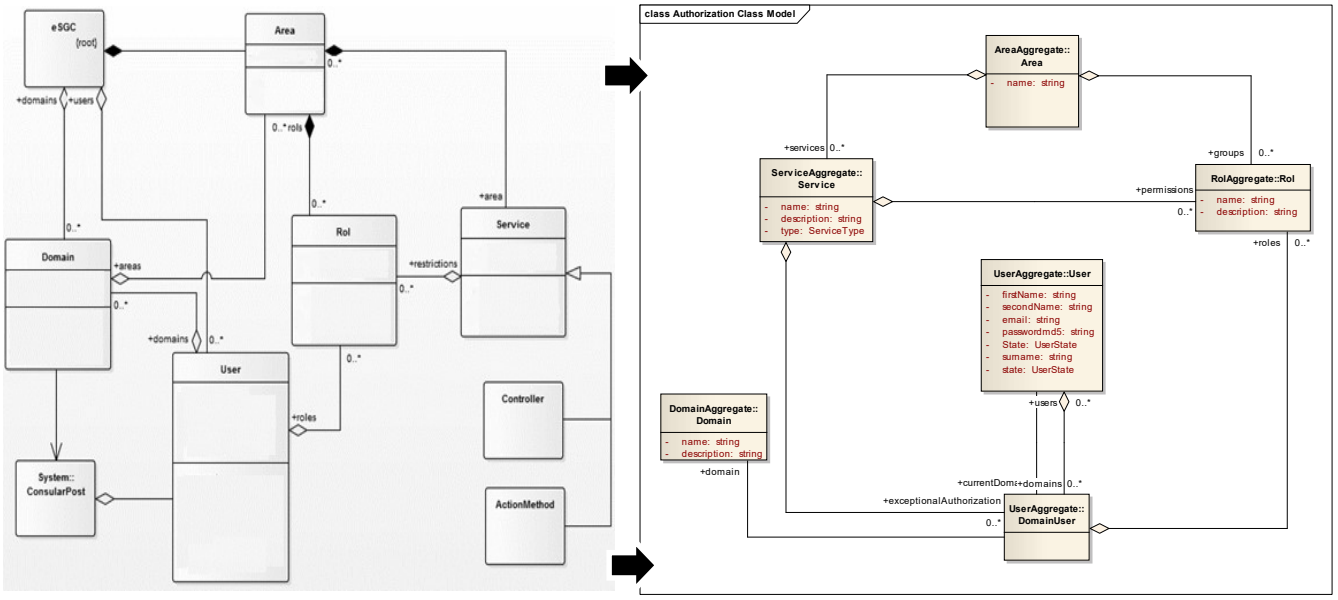


ANNEX 3. DIAGRAMA DE GANTT

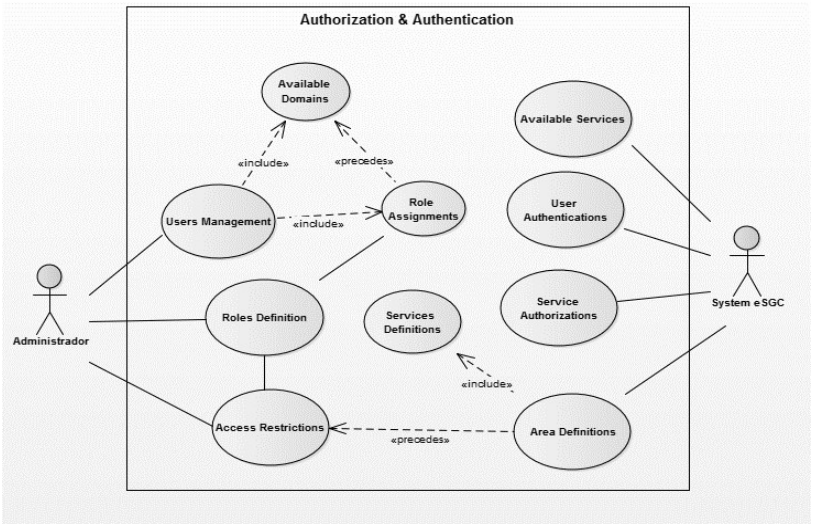


Id	Nombre de tarea	Duración	Comienzo	Fin	Predec
1	Previ: Anàlisi i Especificacions	16 días	lun 01/02/16	sáb 20/02/16	
2	Spring_1: Gestió de la primera entitat al DDD	22 días	lun 22/02/16	mar 22/03/16	1
3	Spring_2: Completar gestió resta entitats	22 días	mié 23/03/16	jue 21/04/16	2
4	Spring_3: Completar codificació basat en el model	20 días	vie 22/04/16	jue 19/05/16	3
5	Spring_4: Desplegament aplicació a plataforma Azure	22 días	vie 20/05/16	lun 20/06/16	4

ANNEX 4. ANÀLISI DE REQUERIMENTS I MODEL ESTÀTIC DE CLASSES D'AUTORITZACIÓ I AUTENTICACIÓ



Modificació Diagrama Classes



Source \ Target	App::Access Restrictions	App::Area Definitions	App::Available Domains	App::Available Services	App::Role Assignments	App::Roles Definition	App::Service Authorizations	App::Services Definitions	App::User Authentications	App::Users Management
RF::Provide actions for eac...				⇐						
RF::Provide authentication ...									⇐	
RF::Provide user authentic...		⇐								
RF::Restrict actions to a Role							⇐			
RF::Restrict Pages	⇐									
RF::Roles Definition						⇐				
RF::System has to able exp...								⇐		
RF::System has to able to a...					⇐					
RF::System has to allow to ...							⇐			
RF::System has to allow to ...									⇐	
RF::Users assigned to offices			⇐							
RF::Users Definition										⇐
RF::Users join to roles					⇐					

Diagrama Autenticació & Autorització i Matriu Casos d'Ús

ANNEX 5. RESULTATS WEB