# Zetes Chronos – Plataforma de mobilitat empresarial

Rubèn Manzano Lomas

**Resum**—Avui en dia les empreses de serveis logístics necessiten una traçabilitat total des de la recollida fins l'entrega, és aquí on Zetes Chronos juga un paper fonamental, ja que treballa cap a l'excel·lència en el servei d'entrega. Per assolir aquesta excel·lència, no només hem de garantir la velocitat d'entrega i recepció dels enviaments en condicions òptimes, sinó que també hem de proporcionar més informació i oferir més opcions i serveis als nostres clients. En aquest projecte s'especifica el procés seguit per a realitzar aquest servei de mobilitat empresarial anomenat Chronos, el qual cobreix totes les necessitats que una empresa de serveis logístics necessita i a més, serà la primera solució software estàndard del grup Zetes Multicom SA, el qual serà desplegat a més de 16 països per la seva adaptació i distribució als nostres futurs clients.

**Paraules clau**— Middleware, Sistema de gestió de transport (TMS), Mapeig objecte-relacional (ORM), MCL, Cloud computing, Front-end, Back-end, traçabilitat, mobilitat, logística, Software com a Servei (SaaS)

**Abstract**— Today the logistics service companies need a total traceability from collection to delivery, there is where Zetes Chronos plays a fundamental role, because it works towards the excellence in service delivery. To achieve this excellence, we not only ensure the speed of delivery and receipt of shipments under optimal conditions, but also we have to provide more information and offer more options and services to our customers. This project specifies the process used to perform this service mobility enterprise called Chronos, which covers all the needs of a logistics service company needs and also be the first software standard solution for Zetes Multicom SA, which will be deployed to more than 16 countries for adaptation and distribution to our future customers.

**Index Terms**— Middleware, Transport Management System (TMS), Object Relational Mapping(ORM), MCL, Cloud computing, Front-end, Back-end, traceability, mobility, logistic, Software as a Service (SaaS

## 1 INTRODUCCIÓ

Zetes és una companyia que cobreix totes les necessitats logístiques que les companyies necessiten, els serveis que ofereixen abarquen desde la gestió en magatzems, fins l'entrega a domicili, a més gestió d'stock de tenda, és a dir, el cicle complet desde que el producte surt de la linea de muntatge fins que arriba al domicili ó magatzem del client. En aquest projecte abarcarem la solució d'entrega a domicili, la qual va desde que el producte surt del magatzem i es carrega al vehicle per procedir a la seva entrega, fins que el client signa la comanda i rep el paquet en qüestió.

En aquest projecte hem innovat molt, gràcies a l'ús de les noves tecnologies que abans no hi èren disponibles, per poder obtenir com a resultat, una solució que tecnològicament vagi alineada amb les futures tendències del mercat, on les necessitats del negoci quedin cobertes al 90% per a tots els clients que puguin necessitar aquest producte i on la gestió de la traçabilitat és total, ja que en cap moment tenim una pèrdua d'informació, la qual cosa ens dona un punt al nostre favor sobre la resta de competidors del mercat. A continuació mostrarem els avantatges que ens aporta tenir aquest exaustiu control de la traçabilitat i de la informació. [Taula I]

TAULA I
Zetes Chronos envers d'altres solucions

| Avantatges |
|---|
| Millor servei |
| Menys errors |
| Visibilitat de la cadena de subministrament |
| Informació en temps real |
| Facilita l'accés a la informació |

En aquest article donarem una visió global sobre l'estat de l'art i els conceptes bàsics necessaris per entendre el conjunt de la solució i veure com els hem aplicat per poder arribar a obtenir els objectius en el plaç establert pel desenvolupament de la solució. Finalment mostrarem els resultats obtinguts, juntament


- *E-mail de contacte: ruben.manzano@e-campus.uab.cat*

- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per: Xavier Otazu (CVC)*
- *Curs 2013/14*

amb les conclusions i les futures idees i necessitats que anirem cobrim en cada una de les futures versions de la solució.

## 2 Estat de l'art

Actualment existeixen molts sistemes pel control de la logística i traçabilitat, els quals ofereixen uns serveis semblants a la nostre solució.

Un exemple d'empreses competidores podria ser Antares Vision, entre d'altres, però no ofereixen un servei integral o bé la unificació de totes les necessitats logístiques, tal i com fem nosaltres, ja que aquest projecte forma part de les altres 6 solucions que completen tota la gestió logística, les quals poden ser contractades com un servei integral o bé adquirir un o varis que necessitem, la qual cosa favoreix que el client tingui tota la seva logística centralitzada en una eina de confiança.

La nostra solució pretén ser un servei excel·lent entre els que hi ha actualment al mercat, garantint la seguretat, traçabilitat, escalabilitat i flexibilitat que els altres serveis no ofereixen.

## 3 Consideracions preliminars

Abans de començar a parlar en l'actual projecte, primer hem de veure perquè Zetes ha apostat per unificar i estandarditzar el seu servei Zetes Chronos, ja que abans de desenvolupar aquesta nova versió estandarditzada, Zetes ja disponia d'altres solucions i/o projectes per aquest servei.

### *A. Versió anterior de Zetes Chronos*

Com abans he comentat, Zetes ja disponia de solucions per donar aquest servei, el problema era que aquestes es dissenyaven únicament pel client que ho demanava, el que coneixem com "Monkey Project", és a dir, un client un projecte. A finals de l'any passat Zetes, va canvia d'estratègia, ja que va veure que aquesta visió de negoci no era la més satisfactòria, ja que tot i tenir seus a més de 16 països, els recursos no eren utilitzats en la mateixa direcció, de manera que per un mateix servei cada seu tenia la seva solució particular que cobria les necessitats que el client en qüestió necessitava i per tant era treball difícilment reutilitzable. Llavors van optar pel que veurem al següent punt, que va ser estandaritzar el seu servei en una única solució, la qual només requeria unes petites adaptacions a l'entorn del client i a les necessitats que no eren considerades estàndard, per tenir el seu servei Zetes Chronos.

### *B. Canvi de mentalitat i esntandarització. El nou Chronos*

Amb aquesta nova visió de negoci, tots els consultors de negoci i els arquitectes van arribar a definir el que a partir d'ara seria l'estàndar de la solució de Chronos, la qual cosa definia tecnologies, regles de negoci i arquitectura de la solució. Durant aquest temps, vaig participar ens numeroses reunions, les quals van donar lloc a la primera definició de la versió estàndar de Chronos per a tot el grup.

### *C. Inici de l'estandarització*

Durant aquest període en el qual es feien les reunions, es va començar a desenvolupar una solució per cobrir totes les necessitats del negoci, les quasl van ser redactades per l'equip de consultors de negoci, però vaig utilitzar l'arquitectura utilitzada en anteriors versions, ja que l'objectiu era cobrir totes les regles de negoci, fins que es decideixi l'arquitectura. Finalment, la reunió amb els arquitectes es va produir, i va ser en aquell moment on vam canviar de forma radical l'arquitectura de la solució, per a oferir un servei totalment Cloud, flexible, escalable, configurable i el més important, estàndar. Amb aquestes directrius vam adaptar les regles de negoci que teniem definides a la solució, per donar amb el que és ara la solució estàndar de Chronos, la qual ara mateix està essent utilitzada a clients com James Hall (Anglaterra), Azkar (Espanya) i d'altres client a Holanda, i s'espera que tot el grup utilitzi aquesta solució com la base per a la solució que serà entregada al client.

## 4 Objectius

En aquesta secció parlarem sobre els objectius d'aquest projecte, des dels objectius principals fins els objectius secundaris com el disseny i implementació del front-end. També veurem la taula que indica l'importància global d'aquests pel projecte[Taula II].

TAULA II
Prioritat dels objectius

| Objectiu | Prioritat |
|---|---|
| A. Definir els requeriments amb l'equip de negoci | crític |
| B. Definir l'arquitecura i tecnologies a utilitzar amb l'equip d'arquitectes | prioritari |
| C. Definir les classes del model | |
| E. Implementar la Business Layer | prioritari |
| F. Implementar la capa de communicació MCL client | prioritari |
| G. Implementar la capa de communicació MCL Mobility Platform | prioritari |
| H. Implementació del Web Portal | secundari |
| I. Maquetació del Web Portal amb template corporatiu | secundari |

*A. Definir els requeriments amb l'equip de negoci*

Un cop teníem totes les regles, necessitats i expectatives dels clients o possibles clients, el que s'ha de fer és reunir aquelles característiques que són necessàries i realitzables, per a cobrir la solució estàndar. Les regles de negoci seguien essent les mateixes, tot i que ara s'havien d'unificar i veure quan i com es realitzarien.

*B. Definir l'arquitecura i tecnologies a utilitzar amb l'equip d'arquitectes*

Dissenyar i definir l'arquitectura per obtenir una solució SaaS, la qual ens doni la flexilibitat, escalabilitat i estandardització necessària per a tot Zetes. També es definiran les tecnologies que s'utilizaran, no només per l'arquitectura, sinó també en el mòduls front-end.

*C. Definir les classes del model*

Definició i disseny del model UML necessari per cobrir els requeriments definits amb l'equip de negoci. També s'escolliran quines d'aquestes classes persistiràn a la nostre base de dades i quines no, depenent de les necessitats.

*D. Implementar Data Access Layer*

Dissenyar i implementar una capa d'abstracció amb una eina ORM per poder mapejar les entitats del model amb la base de dades. A més s'implementarà la part de creació de la base de dades utilitzant mètriques d'optimització i performance de la solució.

*E. Implementar Business Layer*

Dissenyar i implementar la capa de negoci, on s'especificaràn totes les regles de negoci que la nostra solució ha de facilitar.

*F. Implementar la capa de comunicació MCL client*

Dissenyar i implementar la capa de comunicació amb el client MCL per poder enviar i rebre dades de forma síncrona des del client MCL fins el servidor. Totes aquestes peticions es faran via webservices.

*G. Implementar la capa de comunicació MCL Mobility Platform*

Dissenyar i implementar la capa de comunicació amb MCL Mobility Platform per les comunicacions asíncrones. Aquesta capa sempre serà per rebre, mai per enviar, ja que MCL no soporta la recepció asíncrona al client.

*H. Implementació del Web Portal*

Disseny i implementació de la comunicació entre el servidor i web portal com a front-end de la part servidora, a més del disseny i implementació de tota la funcionalitat del web portal.

*I. Maquetació del Web Portal amb template corporatiu*

Disseny i adaptació del web portal per a utilitzar el look and fill corporatiu i satisfer les necessitats i expectatives del client.

## 5 ABORDAMENT DEL PROBLEMA

Per abordar el problema utilitzem el mètode clàssic de divideix i podràs vèncer. Per fer això, el que es fa és modularitzar la solució el màxim possible, per a poder ser reaprofitada en les altres solucions de Zetes. A més d'això, el fet de modularitzar ens garanteix que el codi sigui fàcil de mantenir i més fàcil d'ampliar noves funcionalitats en les següents versions.

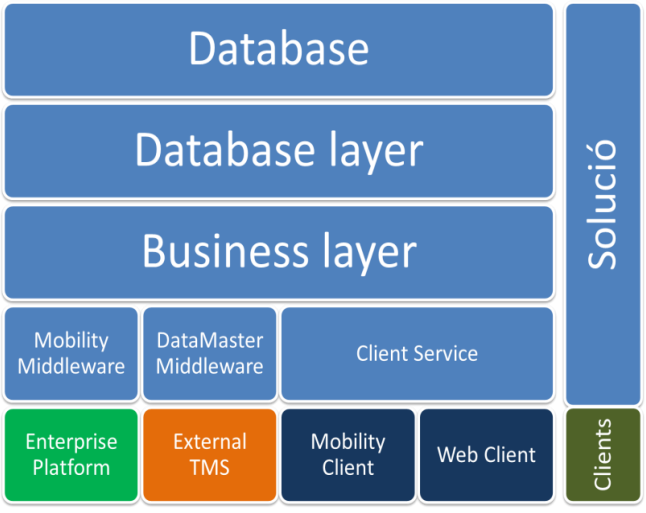


Figura 5.1 Mòduls de Chronos

## 6 METODOLOGIA

El departament d'innovació de l'empresa Zetes Multicom treballa amb la metodologia àgil de desenvolupament SCRUM.

SCRUM és un model àgil de desenvolupament que es basa en un procés iteratiu i incremental. Està enfocat en la gestió de processos de desenvolupament de software.

Els conceptes, característiques i metodologies d'SCRUM aplicades utilitzades han sigut:

*A. Definició del product Backlog*

Inventari de funcionalitats, requeriments i millores basat en les expectatives que té el client. És un conjunt

d'històries valorades en punts.

*B. Requeriments atomitzats*

També coneguts com històries, subdividides en tasques. Una tasca pot ser d'anàlisis, desenvolupament, proves i documentació. Acostumen a ser petites (1-12h). Per a cada tasca es fa una predicció del temps que es trigarà, així com indicar el temps real al final (per poder revisar posteriorment les previsions)

*C. Planificació mitjançant sprints*

Cada 4 setmanes, es realitzen un nombre tancat d'històries. Durant un sprint tan sols es treballa en el que es necessita per l'sprint ("no es contempla el futur")

*D. Reunions*

S'han fer els Daily Scrums diàriament per especificar en què s'està treballant i en qué es vol treballar durant el dia. Aproximadament de 5 minuts.

També s'han realitzat session de valoració per puntuar les histories.

Els Sprints Pre-planning han sigut realitzats abans d'iniciar l'sprint i era on definiem el contingut d'aquest.

Un cop fèiem els Sprints Pre-planning, el que fèiem era l'Spring Planning que era on confirmàvem i tancàvem el contingut final que tindria l'sprint en qüestió.

Un cop acabàvem l'sprint, el que fèiem era un Sprint Review per valorar el resultat de l'sprint superat.

El projecte ha estat realitzat durant els mesos de Febrero a Juny de 2014. Han estat un total de 4 sprints de 4 setmanes cadascún aproximàdament.

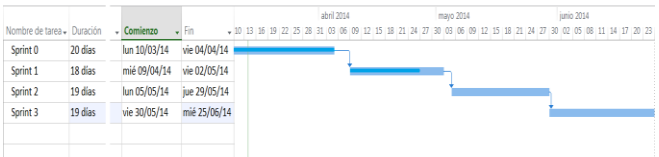


Figura 6.1 Diagrama de Gantt amb els Sprints a realtizar.

Cada sprint consta d'aproximadament 15 punts. Cada punts equival a 5 hores de feina. Les tasques definides per a cada sprint han estat les següents:

TAULA III
Definicio sprints i divisió tasques

| Tasca | Dedicació |
|---|---|
| **Sprint 0** | 15 punts |
| Presa de requeriments | 2 punts |
| Documentació de l'informe inicial | 1 punt |
| Aprenentatge de l'IDE i les tecnologies escollides | 8 punts |
| Implementació d'una part de la Data Access Layer | 2 punts |
| **Sprint 1** | 15 punts |
| Implementació de la Data Access Layer | 8 punts |
| Documentació de l'informe de progrés I | 1 punt |
| Implementació d'una part de Business Layer | 6 punts |
| **Sprint 2** | 15 punts |
| Implementació completa de la Business Layer | 4 punts |
| Implementació de la capa de servei REST | 10 punts |
| Documentació de l'informe de progrés II | 1 punt |
| Definició i implementació del test cases | 2 punts |
| **Sprint 3** | 15 punts |
| Implementació del web portal | 5 punts |
| Definició i implementació dels integration tests | 2 punts |
| Maquetació del web portal | 3 punts |
| Definició i execució dels acceptance test | 2 punts |
| Article final | 3 punts |

## 7 ARQUITECTURA

A l'inici del projecte l'arquitectura escollida per començar a desenvolupar, va ser la típica estructura d'un projecte on tot està centralitzat en una única solució i tots el components formen el projecte, és a dir, el projecte està
compost per una sèrie de mòduls depenents entre sí que donen com a resultat una solució tancada.

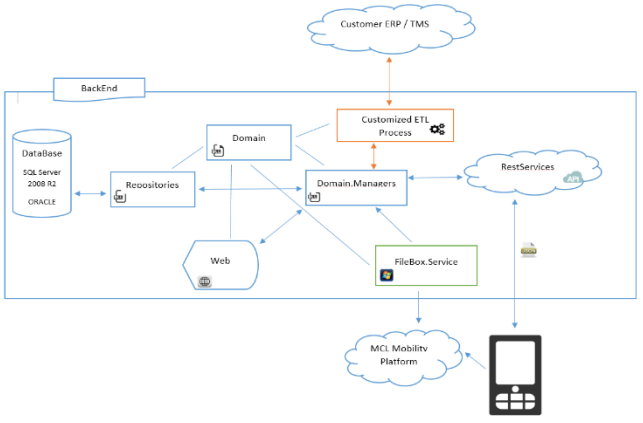


Figura 7.1 Arquitectura inicial de la solució

Quan vam començar a desenvolupar l'aplicació ens vam adonar que la idea de negoci no podia encaixar en aquesta arquitectura, ja que no volíem un projecte, sinó un producte, és a dir, un servei en cloud computing amb mòduls externs que interactuen amb aquest de manera remota i no eran depenents de la solució, sinó que són mòduls totalment desacoplats.

Vam agafar com a exemple un tipus d'arquitectura SOA, per poder tenir idees del que necessitàvem, però després vam adaptar-ho per tenir una arquitectura

totalment modular.

El que vam fer va ser dissenyar un nucli central anomenat BackEnd on hi és tota la lògica de negoci, les dades i la porta d'entrada a les dades mitjançant una capa de comunicació REST. La idea era tenir una única porta d'accés a l'entrada per tots el mòduls, per fer que el mòduls siguin externs a la solució, de manera que podríem oferir una solució en cloud amb mòduls opcionals que es podrien afegir en funció de les necessitats del client.

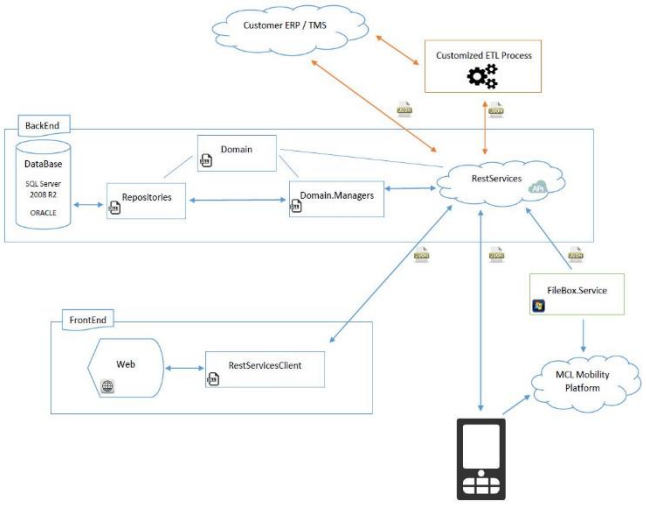


Figura 7.2 Arquitectura final de la solució

Un cop ja teníem dissenyat el core de la solució, vam començar a dissenyar els diferents components que formarien part de la solució base.

## 8 DISSENY DE L'APLICACIÓ: BACK-END

El Back-End és un macrocomponent de la solució, el qual està format per microcomponents. Per a la solució global, aquest macrocomponent és el core, és a dir, és la part més important.

Està format per una capa de comunicació REST, unes biblioteques de clases anomenades Domain, BusinessControllers i Repositories, desenvolupat en C#.

### A. *Microcomponent de comunicació REST*

El RestServices proporciona un bus comú per a tots els components, el qual mitjançant l'intercanvi de dades mitjançant un format json, podem donar servei a tots el nostres components externs o mòduls de forma unificada i estàndar, ja que tot és gestionat per aquest.

### B. *Microcomponent de negoci*

El BusinessControllers és on s'apliquen totes les regles de negoci i tota la lògica de la solució. Està ubicat entre la capa de comunicació REST i l'accés a les dades, per tant és el punt encarregat de servir, proveir i filtrar totes les dades, tant d'entrada com de sortida. A totes les implementacions s'han definit una interfície i s'ha aplicat injecció de codi, després veurem que és això. Ha estat desenvolupat com una DLL (Biblioteca de classes).

### C. *Microcomponent d'accés a dades*

El Repositories és on es defineixen els mètodes d'accés a les dades utilitzant un ORM, en aquest cas: Fluent Nhibernate. Els mètodes d'accés inclouen totes les operacions CRUD (creació, lectura, actualització i esborrat) que es poden fer en les dades del domini de la solució. A més, també es defineix les regles de persistència del domini de la solució. A totes les implementacions s'han definit una interfície i s'ha aplicat injecció de codi, després veurem que és això. Ha estat desenvolupat com una DLL (Biblioteca de classes).

### D. *Microcomponent del domini i àmbit*

El Domain és on es defineixen totes les entitats que seràn persistides en la solució. El domini de l'aplicació són els objectes DAO (Data Access Object) i només existeixen en el macrocomponent Back-End.. Ha estat desenvolupat com una DLL (Biblioteca de classes).

## 9 DISSENY DE L'APLICACIÓ: FRONT-END

Aquest macrocomponent està dividit en dos components: Web, RestServiceClient.

### A. *Component Web*

El component Web és un portal web desenvolupat en ASP MVC 4.5, HTML5, CSS3, JavaScript, JQuery i Ajax. A més utilitzant el template SmartAdmin i els components per la interfície d'usuari Kendo MVC UI.

Aquest portal web és l'encarregat de realitzar o proveir a l'usuari la possibilitat de realitzar les tasques CRUD a la masterdata i poder monitoritzar la work data.

Com que ha estat desenvolupat amb MVC, tenim l'ús del patró MVC, definint com el Model d'aquest, els objectes DTO (Data Transfer Object), més contenidors necessaris per enriquir les dades mostrades a la vista. Els objectes DTO són obtinguts mitjançant les peticions REST dels services que ofereix el BackEnd. Les vistes són les pàgines de visualització. I els controladors són on es gestionen les peticions de la vista i es crida al component de RestServiceClient per obtenir o enviar dades cap al BackEnd.

### B. *Component RestServiceClient*

El component RestServiceClient és el component on s'especifiquen tots el consumidors de serveis rest necessaris per a la capa web.

Hi ha un consumidor per cada controlador especificat a la web, de manera que es crea un bus directa entre el consumidor i la petició generada a la vista.

A totes les implementacions s'han definit una interfície i s'ha aplicat injecció de codi, després veurem que és això.

## 10 DISSENY API DE COMUNICACIÓ PEL MCL MOBILITY PLATFORM

MCL Mobility Platform és un servei en cloud que ofereix al client MCL la possibilitat de que quan aquest no té connexió per realitzar peticions contra el BackEnd, poder enregistrar-les de forma automàtica. Per tant, és el servei que permet gestionar peticions asíncrones entre MCL client i el BackEnd.

Per a que el BackEnd pugui rebre aquests peticions s'ha dissenyat una api de comunicació, la qual engloba desde el REST services definit a la capa de comunicació exclusius per gestionar peticions del client MCL fins al mòdul d'integració encarregat de comunicar-se amb la plataforma de mobilitat i traslladar aquestes peticions generades de forma asíncrona cap al BackEnd.

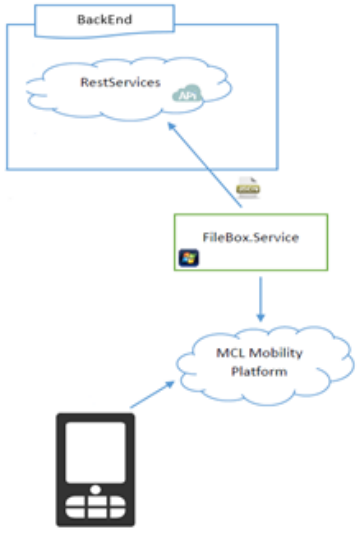


Figura 10.1 Arquitectura API comunicació pel MCL Mobility Platform

No totes les peticions poden ser gestionades de forma asíncrona o síncrona, per tant s'han definit de quin tipus ha de ser cada petició, per exemple: login ha de ser síncrona sempre, en canvi l'actualització de dades d'una workdata sempre serà asíncrona.

El mòdul per obtenir les dades de la plataforma de mobilitat de MCL, és un servei de Windows totalment parametritzable, que el que fa és que cada x segons, en funció dels paràmetres indicats pel client, consulti aquest plataforma per comprovar si hi ha alguna nova notificació per enviar al BackEnd. Per fer això s'han definit uns DTO exclusius per aquesta via de comunicació a més d'uns webservices que gestionen aquestes peticions mitjançant l'enviament de dades en format json per les comunicacions.

## 11 DISSENY API DE COMUNICACIÓ PEL MCL CLIENT

El client MCL simplement és un terminal portàtil que porten els treballadors que utilitzen el sistema Chronos. Està programat sobre una plataforma MCL [Ref]. MCL és un software estàndard per empreses que integra les darreres tecnologies com terminals portàtils, infraestructures inalàmbriques i sistema d'obtenció de dades, basats en lectors de codi de barres, identificació per radiofreqüència i reconeixement de veu.

Aquesta API de comunicaci´p és l'encarregada de donar servei a la comunicació síncrona entre el terminal i el BackEnd. Aquesta API és un disseny basat en REST services definit a la capa de comunicació mitjançant l'enviament i recepció, ja que és bidireccional, de dades en format json.

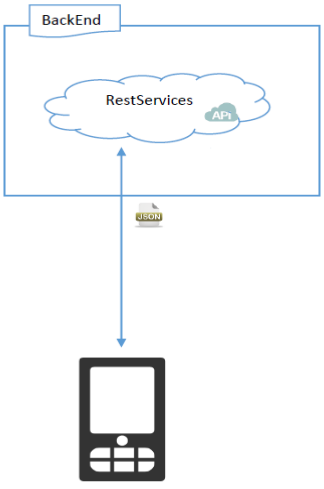


Figura 11.1 Arquitectura API comunicació pel MCL Client

Per a la comunicació entre el terminal MCL i el BackEnd, s'han definit uns DTO específics per cobrir les necessitats del client MCL. A més, la capa de seguretat utilitzada en la resta de capes no ha pogut ser aplicable, ja que el client MCL no soporta HttPBasicAuthentication, l'autenticació per la petició de dades s'ha hagut de dissenyar i crear un nou protocol, mitjançant les dades de l'usuari loguejat i variables de sistema del terminal com per exemple el Serial Number.

## 12 ÚS DE PATRONS DE DISSENY

Per fer un disseny estàndard, flexible i escalable, vam estar molt de temps pensant com podríem dissenyar-ho i une de les coses que teníem clar era l'utilització d'alguns patrons de disseny que ens donéssim aquesta flexibilitat, escalabilitat i estandardització.

Els patrons de disseny utilitzat han estat el "Repository", per la part l'accés a dades, "Singleton" per la gestió de la sessió de l'ORM, el Builder per la creació de dades al BusinessController, "MVC" pel front-end de la web, i el "Dependency Injector" utilitzat tant al BusinessController, com al Repositories, per fer possible una solució estàndard fàcilment modificable sense tocar el core inicial. També s'ha utilitzat el patró "Unit Of Work" (UoW), per donar abstracció entre la solució i l'ORM.

A més tot el disseny de la solució ha estat basat en un paradigma relativament nou anomenat "Code First".

*A.* Repository

El patró Repository consisteix en ser un mediador entre el domini de l'aplicació i les dades que li donen persistència. Això dona la potència de que no només no necessita no conèixer la tecnologia de l'ORM escollit, ni com estàn guardades les dades, sinó que només necessita conèixer els mètodes que faciliten l'accés a les a les dades per poder operar amb elles (tasques CRUD).

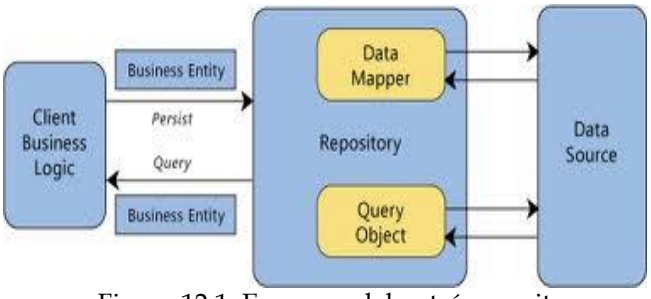


Figura 12.1 Esquema del patró repository

*B.* Singleton

El patró Singleton consisteix en una instància única, és a dir, està dissenyat per restringir la creació d'objectes que pertanyen a una classe o el valor d'un tipus a un únic objecte.

És un dels patrons més coneguts i utilitzats. A més és molt fàcil d'implementar ja que simplement hem d'ocultar el constructor de la classe Singleton, declarar en aquesta classe una variable privada que contingui la referència a la instancia que volem gestionar i després proveir d'una funció en aquesta classe que doni accés únic a aquesta instancia a partir d'aquesta funció o propietat.

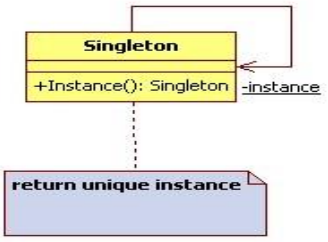


Figura 12.2 Esquema del patró Singleton

*C. MVC*

El patró MVC, Model-Vista-Controlador, és un patró de disseny d'arquitectura de software, que consisteix en separar les dades i la lògica del negoci d'una aplicació de la interfície d'usuari i el mòdul encarregat de gestionar les comunicacions i els events.

MVC es basa en separar aquestes característiques en tres parts diferenciades per funcionalitats, que són el Model que és on hi estàn encapsulades les dades del negoci, la Vista que és la part encarregada de mostrar les dades del negoci i el Controlador, encarregat de gestionar els events i fer d'intermediari entre el Model i la Vista.

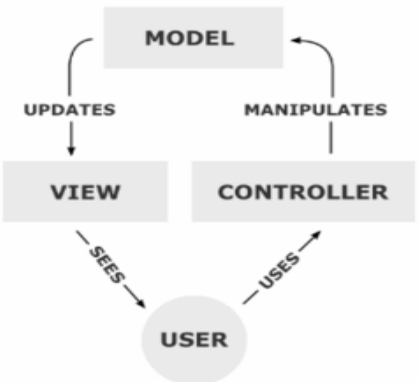


Figura 12.3 Esquema del patró MVC

*D.* Dependency Injector

El patró Dependency Injector, Injector de dependències, és un patró pel disseny orientat a objectes, en el qual s'utilitza objectes de la definició d'una classe (interfície) en comptes de la implementació de la pròpia classe (classe).

Aquest patró el que fa és donar una nivell més d'abstracció, el qual utilitzant un injector de dependències escollir quina implementació vols utilitzar quan es referencia la definició en concret.

Tant C# com Java no suporten aquesta funcionalitat i és necessari l'ús d'un framework per l'injecció de

dependències. En aquest cas s'ha utilitzar Ninject.

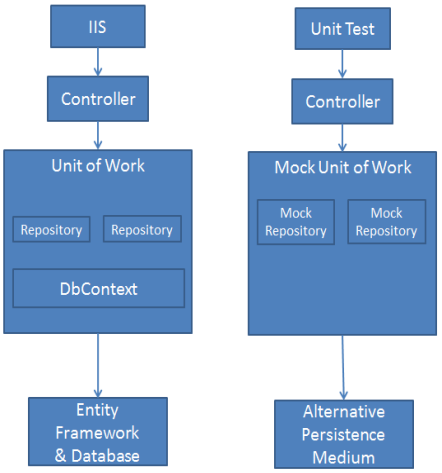


Figura 12.4 Exemple d'esquema pel patró Injector de Dependències

*E. Unit of Work*

El patró Unit of Work, consisteix en que una unitat de treball, manté un registre de tot el que fas durant una transacció comercial que pot afectar la base de dades. En acabar, s'adona de tot el que cal fer per modificar la base de dades com a resultat del seu treball.

Bàsicament el que fa és tractar com a una unitat a tots aquells objectes nous, modificat o esborrats respecte a una font de dades.

El que fa és facilitar un nivell més d'abstracció, seguretat i el més important, ens aïlla de l'ORM escollit per persistir el domini.

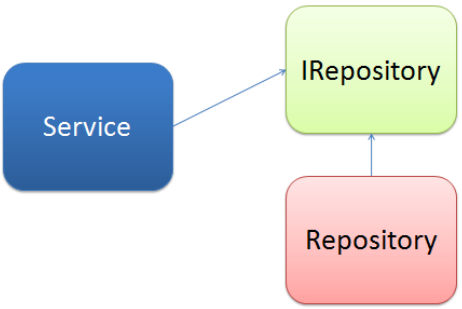


Figura 12.5 Esquema d'exemples utilitzant Unit of Work amb Entity Framework i amb un altre ORM.

*F. Code First*

Code First, és un nou patró de disseny, o bé paradigma, ja que canvia la manera en la qual es dissenya una aplicació.

Code First, consisteix en crear el teu model de classes i després mitjançant unes regles, delegar a un ORM com NHibernate en aquest cas, o bé Entity Framework, per a que persisteixi el teu model de classes i poder generar de forma automàtica la base de dades.

Aquest patró de disseny és relativament nou, i la principal ventatge que proporciona és que no hem de fer res a la nostre base de dades, ja que es crearà automàticament, però el més important, com està basat en el nostre model de classes, com a desenvolupadors, no necessitem conèixer l'estructura de la base de dades, la qual cosa ens ailla a que només hem de conèixer el nostre àmbit de classes per poder manegar totes les dades de l'aplicació.

La principal ventatge respecte al model Database First, és que en aquest cas el model de classes era dependent totalment del model E-R de la base de dades, en canvi amb el nou model Code First, estem totalment aïllats de la base de dades, la qual cosa ens facilita el fet de no tenir que fer ingenieria inversa per crear el model de classes basat en el model E-R de la base de dades, la qual cosa fa que no tinguem cap dependència a l'hora de dissenyar el nostre model de classes.

## 13 TESTING

En l'apartat de testing s'han dissenyat els tests a tots el nivell, desde el unit test per la part del desenvolupador, fins al Integration test per unificar el BackEnd amb els mòduls externs com Front-End, MCL client i MCL Mobility Platform, fins al acceptance test per a la part dels consultors de negoci de la solució Zetes Chronos.

Per la part dels Unit Test, s'ha utilitzat l'eina Unit-Testing, la qual és nativa de Visual Studio per fer testing. Uni test proporciona entre d'altres:

- Afirmacions per verificar resultats.
- Anotacions per definir mètodes de prova.
- Anotacions per definir mètodes addicionals pre i post prova.
- Control d'excepcions, temps d'espera.
- Parametrització de dades.

Els Integration Test, han estat definits i dissenyats per la comunicació entre el BackEnd i els diferents mòduls de comunicació. Part més crítica i on més énfasi s'ha fet ha sigut entre la comunicació del client MCL i la capa de REST services del BackEnd. La utilització d'intercanvi de dades mitjançant format json ha simplificat molt aquesta tasca.

Els Acceptance Test, han estat definits i dissenyats per l'equip de consultors de negoci, els quals van ser executats durant les reunions prèvies al llançament de la primera release de la solució Zetes Chronos.

## 14 GESTIÓ DE LA CONFIGURACIÓ

La gestió de la configuració ha sigut necessària per a controlar i gestionar la versió de desenvolupament, adjuntant millores a la versió release que es feia setmanalment.

Per la gestió de la configuració, s'ha utilitzat el servidor SVN propi (Visual SVN server) amb el que ja està treballant Zetes, però per la part del client s'ha utilitzat Tortoise.

La part més important, va ser la configuració inicial, per determinar quines carpetes i quins fitxers havien d'estar al subversion. El principial problema va ser determinar aquests fixers i carpetes. Un dels problemes que ens vam trobar va ser l'externalització de tots el paquets Nuget en una única carpeta, a més de l'externalització de totes les DLLs dels frameworks necessaris en una altra carpeta. Un cop definit això, es van insertar en el subversions els fitxers i carpetes que formaven la solució a excepció dels bin, obj, .suo i .config, degut a que aquests es generaven automàticament un cop es compilava la solució, per això no havien d'estar al subversion però no causar conflictes a l'hora de que terceres persones es puguesin descarregar la nostre solució.

La release actual i estable sempre estava en el trunk, en canvi, cada persona que volia provar-ho o bé desenvolupar les seves necessitats es creava una branch per al seu desenvolupament. En el trunk només jo tenia permisos d'escriptura, la resta només de lectura, la qual cosa garantia l'integritat de la solució.

## 15 RESULTATS

S'han cobert totes les necessitats de negoci que l'empresa desitjava, és a dir, des de la part de negoci tot ha estat un èxit, ja que s'ha obtingut una solució amb una base estàndard per a tot el grup Zetes Multicom.

Desde la part tècnica s'ha obtingut una solució que tècnicament suportarà la funcionalitat que desde direcció es volia aconseguir amb la creació de solucions estàndard, que és poder obtenir un producte que pugui estar en cloud per la seva distribució. Això ho ha facilitat el fet de que la solució sigui totalment modular, gràcies a que la capa de comunicació per a tots el mòduls de la solució sigui REST service, la qual fa possible aquest nou escenari.

És la primera solució estàndard del grup Zetes Multicom, on formen part més de 1100 treballadors, la qual cosa dona una idea de l'abast del projecte.

La decisió sobre el canvi d'arquitectura que es va prendre gairebé a l'inici del projecte, va ser clau per l'obtenció d'una solució en cloud, la qual ha sigut clau per haver obtingut aquests resultats tant positius per la nova visió empresarial del grup Zetes Multicom.

## 16 CONCLUSIONS

Aquesta solució ha captat l'atenció de moltes persones importants en el grup, degut a que un cop han vist com hem dissenyat, implementat i executat tot el projecte, han vist com la primera solució estandard del grup, ha agradat a tothom i ha obert un nou camí en l'estandardització de la resta de solucions.

Per la part individual puc dir que he aprés més que mai, ja que he tingut la possibilitat de conèixer gent amb molt experiència i poder fer un intercanvi d'opinions i perspectives molt enriquidores per mi.

He aprés com manegar reunions i presentacions amb caps intermitjos, no només en català o castellà sinó que també en anglés, la qual cosa crec que m'ha fet madurar exponencialment com a futur professional en el món de la informàtica.

## 17 TREBALL FUTUR

El treball només acaba de començar, aquesta és la primera release de la solució, però encara s'han d'afegir noves funcionalitats per poder tenir una solució que arribi a totes les necessitats de tots el client que la puguin necessitar, per tant encara queda molt recorregut.

La idea és un cop ja hem establert la base de la solució, l'equip de negoci ens anirà reportant totes aquelles noves necessitats que requereixen els notres clients o bé totes aquelles necessitats que els nostres experts de negoci creuen que poden necessitats, per poder incloure-les dintre de la solució estàndard i que tot el grup es beneficii d'aquesta nova mentalitat de negoci.

## AGRAÏMENTS

El projecte que presento és l'esforç en el qual indirecta o directament vam participar varies persones, ja sigui col·laborant, donant suport o simplement estant amb mi en els moments difícils i en el moments feliços.

Vull agrair a Xavier Otazu per haver escollit el meu projecte i tota la informació, formació i consells que he rebut per part seva, tant durant la realització del projecte com durant aquests darrers anys de la meva formació.

Vull agrair a Jordi Soler per haver sigut el meu mentor i principal defensor meu front decisions importants durant el desenvolupament del projecte i pel fet d'haver-me donat l'oportunitat de formar part del seu equip. També a Victor Quesada per tota la seva col·laboració i ànims rebuts durant el desenvolupament del projecte.

Vull agrair a la Universitat Autònoma de Barcelona per haver acceptat el fet de fer el meu propi projecte a Zetes i per tota la formació rebuda durant aquests darrers anys.

Per últim i més important de tots, vull agrair a la meva família, a la meva mare Conchi, al meu pare Higinio, al meu germà Javi i a la meva parella Jessica, per tot el seu suport, estima i ànims rebuts durant no només el desenvolupament del projecte, ni en aquests darrers anys de la meva formació, sinó durant tota la meva vida.

## BIBLIOGRAFIA

[1] Adam Freeman, « Pro ASP.NET MVC 4 ». Apress, Edition 3, 27 de Decembre de 2012. ISBN: 978-1-4302-4236-9

[2] Armando Fox, David Patterson, «Engineering Software As A Service: An Agile Approach Using Cloud Computing ». Strawberry Canyon LLC, 2nd Edition, 16 d' Abril de 2013. ISBN: 978-0984881246

[3] Jeff Levinson, «Software Testing with Visual Studio 2010». NORTHWEST Cadence, 1ª Edició, 3 de Març de 2011. ISBN: 978-0321734488

[4] Martin Fowler, « Patterns of Enterprise Application Architecture. ». Back Cover, 1ª Edició, 15 de Novembre de 2003. ISBN: 978-0321127426

[5] Jess Chadwick, Todd Snyder, Hrusikesh Panda, «Programming ASP.NET MVC 4 ». O'Reilly Media, 1ª Edició, Septembre de 2012. ISBN: 063-6-920-02404-0

[6] Benjamin Perkins, «Working with NHibernate 3.0 ».Wrox, 1ª Edició, 25 d'Agost de 2011. ISBN: 978-1-118-11257-1

[7] Jason Dentler, «Nhibernate 3.0 Cookbook ». PACKT PUBLISHING, 1ª Edició, 4 d'Octubre de 2010. ISBN: 9781849513043

[8] Sam Guckenheimer, Neno Loje, «Agile Software Engineering with Visual Studio: From Concept to Continuous Feedback». Ken Schwaber, 2ª Edició, 23 de Septembre de 2011. ISBN: 978-0321685858

[9] Craig Strong, A Really Simple Explatanion Of MVC, Març de 2013. http://c2.com/cgi/wiki?ModelViewController