DESIGN AND DEVELOPMENT OF AUDIOVISUAL TELEEDUCATION SERVICES OVER THE INTERNET

Final Report

by Anna M. Carné Vilà

Project supervisor: Dr. Jorge Mata Diaz

Barcelona, July 2013
Revision history

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23/06/2013</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>2/07/2013</td>
<td>First Review of Core content</td>
</tr>
<tr>
<td>2</td>
<td>12/07/2013</td>
<td>Complete document</td>
</tr>
<tr>
<td>3</td>
<td>19/07/2013</td>
<td>Complete review</td>
</tr>
<tr>
<td>4</td>
<td>21/07/2013</td>
<td>Final document</td>
</tr>
</tbody>
</table>

Document distribution list

<table>
<thead>
<tr>
<th>Name</th>
<th>E–mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna M. Carné Vilà</td>
<td><a href="mailto:anna.carne.vila@gmail.com">anna.carne.vila@gmail.com</a></td>
</tr>
<tr>
<td>Jorge Mata Diaz</td>
<td><a href="mailto:jmata@entel.upc.edu">jmata@entel.upc.edu</a></td>
</tr>
</tbody>
</table>

Agreement consent

<table>
<thead>
<tr>
<th>WRITTEN BY:</th>
<th>REVIEWED AND APPROVED BY:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>20/07/2013</td>
<td>21/07/2013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna M. Carné Vilà</td>
<td>Jorge Mata Diaz</td>
</tr>
</tbody>
</table>
Abstract

The present work arises from the need of a company, in this case the student’s family business, to provide services over the Internet. The company is a Yoga school, where currently a number of courses on different yoga-related topics are taught.

Based on this need and the knowledge acquired during her studies, the student comes up with the idea of implementing an e-learning platform, able to distribute audiovisual contents in order to practice and learn Yoga at distance. In this regard, she poses the idea to the teacher of the subject Comunicacions Multimèdia, taken the previous year. They analyzed together the problem and elucidate a number of possible solutions.

The project is focused on two fundamentals aspects: endowing the platform with an added value on the service and taking into account scalability issues.

On the one hand, after the analysis of some existing platforms, we will give as an added value the possibility of maintaining conversations among the teacher and students in order to resolve doubts and motivate them.

On the other hand, popular web services may suffer congestion and bottlenecks due to large number of requests they receive. This problem aggravates even more when the host server upload bandwidth is not excessively high. Additionally, society has progressed so far with connectivity that anything less than instantaneous, on-demand communication and content is considered a problem. In this regard, we propose to make use of CDNs, in order to scale the service by means of the use of caches and DNS advanced techniques.
Resum

Aquest treball sorgeix de la necessitat que té una empresa, en aquest cas l’empresa familiar de l’estudiant, d’estendre el seu negoci per fer-lo visible a Internet. El negoci és una escola de Yoga, on actualment s’imparteixen multitud de cursos en diferents temàtiques relacionades amb el Yoga.

Partint d’aquesta necessitat, i tenint en compte els coneixements adquirits a la carrera, l’estudiant pren la iniciativa d’implementar una plataforma d’educació on-line, on es puguin servir continguts audiovisuals que permetin practicar i aprendre Yoga a distància. Així doncs, l’estudiant trasllada la idea al professor de l’assignatura Comunicacions Multimèdia, realitzada el quadrimestre anterior. Junts, analitzen el problema i plantegen una sèrie de possibles solucions.

El projecte es centra en dos aspectes fonamentals: dotar la plataforma d’un valor afegit en el servei i tenir en compte l’escalabilitat del sistema.

D’una banda, després de l’anàlisi d’algunes plataformes existents, com a valor afegit es donarà la possibilitat de mantenir converses amb el professor per resoldre dubtes i motivar als alumnes.

D’altra banda, els serveis web més populars poden patir problemes de congestió i colls d’ampolla degut al gran nombre de peticions que reben. Aquest problema s’agreuja encara més quan l’ample de banda de pujada del servidor de continguts no és exessivament alt. A més, l’evolució de la connectivitat que hi ha hagut els últims anys fa que tot allò que no és instantani sigui vist com un problema per part dels usuaris. En aquest sentit, es proposa l’ús de xarxes de distribució de contingut. Aquestes xarxes permeten fer el servei escalable, mitjanant l’ús de caches i tècniques avançades de DNS.
Resumen

Este trabajo surge de la necesidad que tiene una empresa, en este caso la empresa familiar de la estudiante, de extender su negocio a Internet. La empresa es una escuela de Yoga donde actualmente se imparten multitud de cursos en diferentes temáticas relacionadas con el Yoga.

Partiendo de esta necesidad, y teniendo en cuenta los conocimientos adquiridos durante la carrera, surge la idea de implementar una plataforma de educación on-line donde se puedan servir contenidos audiovisuales que permitan practicar y aprender Yoga a distancia. Así pues, la estudiante traslada la idea al profesor de la asignatura *Comunicacions Multimèdia*, realizada en el cuadrimestre anterior. Juntos analizan el problema y plantean una serie de posibles soluciones.

El proyecto se centra en dos aspectos fundamentales: dotar la plataforma de un valor añadido en el servicio y tener en cuenta la escalabilidad del sistema.

Por un lado, después del análisis de algunas plataformas existentes, como valor añadido se dará la posibilidad de mantener conversaciones con el profesor para resolver dudas y motivar a los alumnos.

Por otro lado, los servicios web más populares pueden sufrir problemas de congestión y cuellos de botella, debido al gran número de peticiones que reciben. Este problema se agrava aún más cuando el ancho de banda de subida del servidor de contenidos no es excesivamente alto. Además, la evolución en la conectividad que ha habido los últimos años hace que todo aquello que no sea instantáneo sea visto como un problema por parte de los usuarios. En este sentido, se propone el uso de redes de distribución de contenido. Estas técnicas permiten escalar el servicio mediante el uso de caches y técnicas avanzadas de DNS.
Agraïments

En aquestes pàgines he intentat plasmar el treball d’aquests últims 4 mesos, uns mesos que s’haguessin fet eterns sense vosaltres. Però no només això, representa el final d’una etapa i l’inici d’una nova on espero que tots vosaltres em seguiu acompanyant com fins ara.

M’agradaria agrair primerament, al meu director Jorge Mata que des del primer moment va creure en la meva idea i va tirar endavant amb ella. Sense ell res d’això hagués estat possible.

_Muchas gracias!

A les nenes de secretaria que vam estar juntes molts anys i sempre em van recolzar i ajudar en tot el que vaig necessitar. Especialment a l’Ana Benedicto que s’ha portat sempre increïblement amb mi.

_Us dec una recepta!! Sou les millors

A les meves amigues de la uni que tot i haver anat molt atrafegades totes, aquest quadri, han compartit aquesta experiència amb mi durant aquests gairabé 6 anys. Els hi desitjo el millor.

_Nenes!!Que dintre de poc ja acabeu vosaltres també!!

A la Mecè Roy, la meva iaia postissa que essent una desconeguda hem va acollir a casa seva com si fós la seva neta i em va enseñar a portar una casa.

_Sempre et portaré al meu cor

v
Als meus amics de Vilanova que han estat molt comprensius amb la meva falta de temps per dedica’ls-hi. I s’han interessat sempre per mi i el projecte. Gràcies per dedicar una estoneta de la nit de Sant Joan a fer simulacions amb mi.

Sou genials i SEMPRE em tindreu al vostre costat.

Especialment vull donar les gràcies a la Karina Ferrando que sabent el poc temps que tenia venia ni que fós 1h a casa per veure’m, posar-nos al dia i ajudar-me amb el que fes falta.

T’estimo Kari, saps que sempre em tindrás

A Toni Torrellas, Consol Socastro, Sergi Torrellas i Núria Matías per acollir-me a la seva família i fer-me sentir sempre la seva presència al meu costat i els seus ànims constants.

Gràcies per estar incondicionalment al meu costat

A la meva germana Rosa Carné per les nits en vela parlant des del cor.

Gràcies per encomanar-me sempre la teva illusió, les teves ganes de viure i superació i sobretot el teu amor

A la meva mare, M. Rosa Vila per estar sempre a l’altre banda del telèfon per tot el que fes falta. Per preocupar-se sempre per mi en totes els aspectes i ajudar-me en el dia a dia. I a la seva parella Narcís Pascual per fer-la tan feliç.

Gràcies Mama, sóc feliç de veure com has refet la teva vida i jugues a nines. T’estimo

I per anar acabant, vull donar les gràcies al meu pare Lluís Carné que durant aquests 5 mesos ha estat al meu costat cada dia ajudant-me a tirar endavant en els pitjors moments, fent mil viatges per estar al meu costat i ajudar-me quan més ho he necessitat. Mai li podré agrair prou el que ha fet i està fent per mi. A part de ser el propulsor d’aquest projecte que sense el seu interès i curiositat pel món de la tecnologia no hagués estat possible.
Papi, ets el meu creador, el meu guia, mi lugar en el mundo.

Finalment, vull donar les gràcies a una de les persones que viatja al meu costat en aquest camí que és la vida, la meva parella Marc Torrellas per haver estat tan comprensiu amb mi aquests mesos de tanta feina, haver-me ajudat en tot el que ha pogut i no deixar mai que s’apagués la llum.

*El sol no surt si tu no hi ets.*

*Amb tu ser feliç és com respirar, INEVITABLE*
Contents

List of Figures x

Acronyms xi

1 Introduction 1
  1.1 Motivation .............................. 1
  1.2 Goals ................................... 2
  1.3 Organization ............................ 3

2 Time plan 5

3 System description 7

4 Web server implementation 11
  4.1 Web server functionalities ................. 11
  4.2 Utilised technologies ..................... 15
  4.3 Apache and CI configuration ............... 22
  4.4 Web implementation ...................... 23

5 Content Delivery Network 36
  5.1 Introduction ............................ 36
  5.2 Coral CDN ................................ 39
  5.3 Protocols used ............................ 42
  5.4 Simulations ................................ 43
  5.5 Conclusions .............................. 48

6 Budget 49
  6.1 Development phase ....................... 49
  6.2 Test phase ................................ 50
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3 Maintenance phase</td>
<td>52</td>
</tr>
<tr>
<td>6.4 Enhancement phase</td>
<td>53</td>
</tr>
<tr>
<td>7 Conclusions</td>
<td>54</td>
</tr>
<tr>
<td>A Code</td>
<td>56</td>
</tr>
<tr>
<td>A.1 Controllers</td>
<td>56</td>
</tr>
<tr>
<td>A.2 User model</td>
<td>59</td>
</tr>
<tr>
<td>A.3 Views</td>
<td>60</td>
</tr>
<tr>
<td>B Budget tables</td>
<td>70</td>
</tr>
<tr>
<td>B.1 First year</td>
<td>70</td>
</tr>
<tr>
<td>B.2 Second year</td>
<td>72</td>
</tr>
<tr>
<td>B.3 Third year</td>
<td>74</td>
</tr>
<tr>
<td>Bibliography</td>
<td>76</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Time plan updated .............................................. 6
3.1 Initial Scenario ............................................... 7
3.2 CDN Scenario .................................................. 8
3.3 Client-server interaction ...................................... 9
4.1 Home page ...................................................... 12
4.2 Register a new user ........................................... 13
4.3 Thank you for registration .................................... 13
4.4 Log in ........................................................... 14
4.5 Chat room ....................................................... 14
4.6 Log out button and started session ......................... 15
4.7 MVC Architecture ............................................. 19
4.8 System directories structure ................................ 21
4.9 MVC implemented ............................................. 24
4.10 Log in functionality ......................................... 27
4.11 Log out functionality ........................................ 29
4.12 Sign in functionality ......................................... 30
4.13 Chat functionality ........................................... 34
5.1 Typical network vs CDN ...................................... 37
5.2 Basic interaction flows in a CDN environment ........... 37
5.3 Steps to serve a Coralized URL ............................... 40
5.4 HTTP GET video request ..................................... 44
5.5 Flow graph of the TCP streams .............................. 44
5.6 IO Graph ........................................................ 45
5.7 IO Graph for $N_{\text{users}} = 3$ ................................. 46
5.8 HTTP GET video request using CoralCDN ................ 47
5.9 Diferent IPs when using CoralCDN ......................... 48
5.10 Flow graph of the TCP streams when using CoralCDN .. 48
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ajax</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>BW</td>
<td>Bandwidth</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Delivery Network</td>
</tr>
<tr>
<td>CFS</td>
<td>Cash Flow Statement</td>
</tr>
<tr>
<td>CI</td>
<td>Code Igniter</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>International Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPC</td>
<td>Índice de Precios al Consumo</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>PL</td>
<td>Profit and Loss</td>
</tr>
<tr>
<td>ROE</td>
<td>Return on equity</td>
</tr>
<tr>
<td>RTSP</td>
<td>Real Time Streaming Protocol</td>
</tr>
<tr>
<td>SEO</td>
<td>Search Engine Optimization</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WWW</td>
<td>Wolrd Wide Web</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Over the last few decades, the Internet has made a revolution in our society and our economy. Driven by the emergence of the World Wide Web (WWW), the Internet has been used not only as a mechanism for information dissemination and broadcasting, but also for business purposes. The Web shapes a universe of information accessible via networked computers, offering content in the form of Web pages, images, texts, animations, or audio/video streams that can be offered to users.

Electronic Commerce (e-commerce) is the buying and selling of products and services over electronic systems such as Internet. For self-created and non-digitizable goods, Internet has opened a window enabling retailers to sell to anyone, anywhere and at any time. The advantages of selling online helped many for buyers and sellers to: open 24/7, enlarge geographical reach, outcome cheaper prices and costs, wider variety of services, etc. These reasons, together with the increasing penetration of high-speed and mobile Internet, safe payment and adoption of new consumer habits have made e-commerce to raise in the last years through the entire world.

The project arises from a yoga company owner who wants to make use of the Internet to extend their services, currently served locally by means of in-person classes. The interest of this resides on three fundamental aspects. First, make visible the business to the Internet may attract new clients if online marketing is efficiently exploited. Second, customers could be interested on practicing yoga at home, for some different reasons, e.g. save time, privacy, etc. And finally, yoga is a discipline that might make people feel better or heal them. Therefore, it is interesting to offer the service at any time and at any place in order to let customers using the service when they really need, e.g. due to insomnia periods.
1.2 Goals

The service will be offered as a collection of videos organized as yoga courses composed of a number of lessons. The project aims to develop an e-learning platform over the Internet able to provide the lessons via video streaming.

The streaming technology is often thought of as the playback of continuously flowing media such as audio and video. A more accurate description, however, considers the distinction between true streaming technology and the simple playback of downloaded audio or video files. Prior to its invention, users had to download audio and video files in their entirety before starting playback. This is usually not a problem with relatively small text documents or images, either of which can be downloaded very quickly. The large sizes of audio and video files, however, generally translate into painfully long download delays before playback begins. Streaming technology addresses this problem by establishing a permanent data flow from the server to the client, allowing the client to listen or watch the content meanwhile it is downloaded. Consequently, it is no longer required to fetch the entire audio or video file before playback starts, which significantly reduces the initial playback delay, but it is not enough.

Popular Web services may suffer congestion and bottlenecks due to large demands made on their services, even more when the host server upload bandwidth is not excessively high. Such a scenario may cause unmanageable levels of traffic flow, resulting in many requests being lost or with unacceptable long delays. A possible way to overcome this issue is using a cache system, either at the client or the server, useful because it stores what the client requests in a way such that accessing to this data again results faster.

In this scenario is where CDNs emerge to overcome the inherent limitations of the Internet. A Content Delivery Network (CDN) replicates content from the origin server to cache servers, scattered over the globe, in order to deliver content to end-users in a reliable and timely manner from nearby optimal surrogates.

1.2 Goals

This work is focused on the design of a platform able to provide e-learning initiatives via the Internet. The business consists on offering courses, in turn divided in lessons, to be watched by clients. To this end, clients can purchase each lesson or the full course and watch them a finite number of times, after which they have to pay again. The novelty of this project with respect to other e-learning courses is the possibility of consulting to the teacher in order to explore in depth a particular topic or resolve doubts. Moreover, every week some in-person classes will be recorded and uploaded
to the site, where some of them will be free. The aim of those classes is to attract new clients, retain the current clients, update the site periodically and for viral marketing purposes. Finally, it is expected that once a month (or twice depending on the demand) a real-time lesson might be imparted, in order to motivate the students and create a closer environment.

In this context, we identify some tasks to be tackled, summarized next:

1. To identify users accessing to a web page by means of a registry and log in/log out procedures. This enables service differentiation between the visitors and registered clients.

2. To implement a shop chart managed with secure payment procedures\(^1\), such that the client is able to purchase contents according to data protection laws.

3. To implement real-time communications by means of a chat or videoconference\(^2\) applications, enabling the possibility of improving the quality of experience of the e-learning courses.

4. To provide pre-recorded classes via the Internet

5. To provide concurrent access of students. In particular, it is expected that high bursty traffic flows demand audiovisual contents, thus our system should be able to manage them.

6. To develop a budget so as to evaluate the costs, initial investment and financial feasibility related to the present project.

1.3 Organization

The present work is organized as follows:

- First, chapter 3 describes the topology of the system, starting from current architecture and presenting the proposed solution using CDNs. Moreover, an overview of how the implemented and/or used entities interact in the client-server communication is presented.

- Chapter 4 is focused on the implementation of the Web server. We explain first the developed site from a user point of view, and next how all these functional-

\(^1\)This functionality will be implemented in future work

\(^2\)For simplicity, we implement the chat application, while the design of a videoconference can be tackled similarly.
ties are implemented, giving also a review of the employed technologies. Codes are deferred to Appendix A.

- Likewise, chapter 5 describes the CDN system used and how we adapt our Web Server in order to use it. After this, some simulations show the system working with and without the CDN.

- The expected costs and earnings are elucidated in chapter 6, where we develop a budget taking into account the marketing plan, basically based on Google Adwords. All these data will be merged into annual balance sheets, deferred to Appendix B.

- Finally, conclusions and future work are drawn in chapter 7.
Chapter 2

Time plan

In order to organize the work, we designed a Time plan, shown in the *Project Proposal and Work Plan* attached document at the beginning of the project. During the development of this project, such Time plan suffered some modifications, as it can be seen in the *Critical Review document*. The final version is depicted in Fig.2.1.

In first place, we expected that tasks related with the CDN would be easier than were since the initial purpose was to use it and make some simulations. However, the free CDN finally used needs to be implemented, thus it takes a bit of time to understand how it works and how we should modify our site so as to use it.

Secondly, we include the task of developing a Budget for this document. We thought that there would be no time to finish this task, so we did not include it in previous versions. Finally, we have considered it as an important part of the project and, consequently, we included it to the project.

And to conclude, we have modified the dates according to the new deadlines of Final Report and presentation, agreed with the school.
<table>
<thead>
<tr>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research information</td>
<td>14 días</td>
<td>mar 19/02/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>Client analysis</td>
<td>14 días</td>
<td>mar 19/02/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>HTML5</td>
<td>14 días</td>
<td>mar 19/02/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>PHP/JS</td>
<td>14 días</td>
<td>mar 19/02/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>HTTP Streaming protocol</td>
<td>14 días</td>
<td>mar 19/02/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>Developing the work site</td>
<td>50 días</td>
<td>mar 05/03/13</td>
<td>mar 23/04/13</td>
</tr>
<tr>
<td>Understand and adapt a HTML5 and CSS</td>
<td>4 días</td>
<td>mar 05/03/13</td>
<td>vie 08/03/13</td>
</tr>
<tr>
<td>Host, domain and server</td>
<td>1 día</td>
<td>sáb 09/03/13</td>
<td>sáb 09/03/13</td>
</tr>
<tr>
<td>Implement PHP Framework</td>
<td>8 días</td>
<td>dom 10/03/13</td>
<td>dom 17/03/13</td>
</tr>
<tr>
<td>Play a video using HTML5 features</td>
<td>7 días</td>
<td>lun 18/03/13</td>
<td>dom 24/03/13</td>
</tr>
<tr>
<td>Implement a chat service</td>
<td>20 días</td>
<td>lun 25/03/13</td>
<td>sáb 13/04/13</td>
</tr>
<tr>
<td>Implement an intranet to register users</td>
<td>10 días</td>
<td>dom 14/04/13</td>
<td>mar 23/04/13</td>
</tr>
<tr>
<td>CDN analysis and selection</td>
<td>53 días</td>
<td>mier 24/04/13</td>
<td>sáb 15/06/13</td>
</tr>
<tr>
<td>Interaction and architecture of the network</td>
<td>3 días</td>
<td>mier 24/04/13</td>
<td>vie 26/04/13</td>
</tr>
<tr>
<td>Choose and implement a CDN according to the previous information</td>
<td>19 días</td>
<td>sáb 27/04/13</td>
<td>mier 15/05/13</td>
</tr>
<tr>
<td>Simulations</td>
<td>31 días</td>
<td>jue 16/05/13</td>
<td>sáb 15/06/13</td>
</tr>
<tr>
<td>Develop a budget</td>
<td>10 días</td>
<td>dom 16/06/13</td>
<td>mar 25/06/13</td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project proposal and work plan</td>
<td>0 días</td>
<td>lun 04/03/13</td>
<td>lun 04/03/13</td>
</tr>
<tr>
<td>Critical review</td>
<td>0 días</td>
<td>lun 29/04/13</td>
<td>lun 29/04/13</td>
</tr>
<tr>
<td>Final review</td>
<td>0 días</td>
<td>lun 22/07/13</td>
<td>lun 22/07/13</td>
</tr>
<tr>
<td>Public presentation</td>
<td>0 días</td>
<td>vie 26/07/13</td>
<td>vie 26/07/13</td>
</tr>
</tbody>
</table>

Figure 2.1: Time plan updated
Chapter 3

System description

Nowadays, the typical scenario for client-server communications is the one depicted in Fig. 3.1. All user requests for a particular Web page are handled by a single Web server storing the requested content. The problems that arise from this topology are twofold.

Figure 3.1: Topology for a centralized Web Server
On the one hand, this approach has serious scalability problems. The load on the Web Server and on the network link connecting the server to the Internet increases with the number of user requests. This is not a problem for specialized Web sites serving only a small number of interested parties. But as we expect to design a popular Web site (and a successful business), it can easily get overwhelmed with a large number of incoming user requests. This is the case when more and more users request content from a single Web server: either the server’s processing capacity or the bandwidth available on its connection to the Internet can easily be exceeded. If this happens, user requests are dropped, which results in increased access delays or even unavailability of the Web site. Consequently, this approach cannot efficiently work with several traffic flows.

On the other hand, the centralized Web service model relates to the distance between a Web server and potential Web clients. This distance can be measured using different metrics such as number of hops, delay, packet loss on the path, or even geographic distance. Even if a Web server has enough resources to handle all incoming requests in a timely manner, the distance between server and clients can lead to noticeable delays.

![Topology when a CDN service is deployed](image_url)
So far we have described the two major problems of the centralized approach: scalability and distance between server and clients. These two issues can be alleviated by means of using a CDN deployment, as shown in Fig.3.2. The idea behind this approach is the so-called “divide and conquer”. A centralized architecture is much easier to setup and to manage, whereas the distributed architecture comes at the cost of increased complexity and higher initial investment. However, it scales better for large numbers of global users and provides better performance and reliability.

To do so, incoming requests first pass through a front-end load balancer. This component dispatches requests to one of the surrogate servers based on certain metrics, such as the current server load. In turn, surrogate servers refresh, distribute and move the content closer to the user. Therefore, the requested Web objects are stored in an intermediate location between the object’s origin server and the client, and can be served from there. Consequently, the access time can be reduced and significant network resources can be saved—namely bandwidth and processing resources. This architecture is able to solve goal 5 described in the previous section. Now we focus on the communication between each Surrogate server and the clients, as shown in Fig.3.3. At the left-hand side, we observe how the clients send HTTP requests to the server by means the browser. Those requests are managed by the HTTP Server, who is able to perform different actions according to the received request.

In our case we use Apache Web Server [1], not only since its popularity, but also because it supports a variety of features, many implemented as compiled modules which extend the core functionality. In this regard, our PHP scripts will be initiated by the PHP module of Apache in order to accomplish the different functionalities. When necessary, those PHP scripts will communicate with a Database (DB) by means of SQL statements in order to retrieve/store content as a result of the received request.
In the following, we describe as an example how the chat functionality is tackled by the system. First, consider that one of the clients wants to send a message. To this end, the browser sends a request containing such message to the Server. This request is managed by Apache, who executes a PHP script to store the message on the database. On the other hand, at the client side an Ajax script polls periodically the Web Server asking for chat updates. As a result, Apache executes a PHP script to retrieve the talk from the database, and broadcasts it to all connected users.
Chapter 4

Web server implementation

The current implementation includes a chat room, user registration and video streaming. Section 4.1 shows those services working from a user point of view providing an overview of the functionalities. Next, in section 4.2, the technologies used in implementation are briefly introduced whereas configuration and implementation of the web server Apache and the PHP framework CI are explained in section 4.3. Finally, section 4.4 represents the cornerstone of this chapter, where the registry and chat implementation are addressed.

4.1 Web server functionalities

Users are enabled to perform different actions when they access to the page according to their interests. When the user is connected to our site, s/he is provided with diverse information such company profile, courses offered, news, as we can see in Fig.4.1.

The user is also allowed to register himself, as shown in Fig.4.2, which gives the possibility to maintain a chat with the teacher. In future, we plan to implement a payment platform to purchase courses, thus the user will register for the first time when he decides to buy his first course or solve some doubts about the contents.
4.1 Web server functionalities

Figure 4.1: Possible actions at the home page
4.1 Web server functionalities

Figure 4.2: Register a new user

This page can be accessed from any view by selecting the option *No estas registrado?*. Once it is correctly completed (the data will be checked to have the correct format), the page in Fig. 4.3 is shown in order to give advice to the user that the registry has been succeeded. After this action, users are able to log in to the page.

Figure 4.3: Thank you for registration
As before, the client may log in from any view by means of the form in the menu bar, as shown in Fig. 4.4:

![Log in](image1)

**Figure 4.4:** Log in

We expect to implement a private area where frequent customers can access when they log in. In such area, users are able to see all their bought courses, as well as purchasing new courses or modify personal information. In the current implementation, when the user log in is redirected to the chat room, see in Fig. 4.5:

![Chat room](image2)

**Figure 4.5:** Chat room

Finally, when a user wants to leave he can log out. As the other actions it can be done from any view by selecting the option Logout, as shown in Fig.4.6:
4.2 Utilised technologies

The web server uses a wide set of technologies in order to act as explained in the previous section. It has been used a client-server architecture, where some actions are carried out by the browser’s client and others by the web server. Specifically, for the client side we have used HTML5 and CSS to present and structure the information and Javascript and Ajax to interact with the user dynamically for all actions that can be carried out using the client permissions. This means that, for instance, Javascript has not been used to modify the database for security reasons.

On the server side, we used the database manager MySQL, PHP to code the server actions and Apache as the web server. All of them are briefly described next.

4.2.1 HTML5

Hypertext Markup Language (HTML) is a markup language for describing web pages, independent from electronic devices, systems and applications. The HTML language was originally specified by Tim Berners-Lee at the beginning of the 1990s. Since the first version, it has been developed and extended far beyond its initial form. Standardization of HTML was initially moved into the International Engineering Task Force (IETF) and is nowadays carried out by the World Wide Web Consortium (W3C). The latest release is HTML5, launched approximately in 2008, but not actually implemented in most of the browsers until 2011. However, the W3C is currently in development of more robust updates, in order to contribute to the first official recommendation, expected for 2014.

In the beginning, HTML was used to define the layout and the formatting of a Web page, allowing the authors to embed hyperlink references to other resources on the Web. HTML5 arises as an observation of typical usage on modern websites. HTML5 was created to make the coding process easier and logical, but also considering the progressively spread of low-powered terminals, e.g. smartphones or tablets. Some of the new features introduced with HTML5 are explained below.
New structure

The main novelty that HTML5 provides is the new structure for developing websites. Now, the language includes tags based on typical modern websites structure. Some of them are described below:

- `<nav>`: this tag represents the typical part common to all pages of the site devoted to link those different pages, usually called the navigation menu.

- `<figure>` and `<figcaption>`: two tags to manage images

- `<section>`, `<header>` and `<footer>`: the former is used to separate the different sections of the page, whereas the other two can be used to define a header and a footer in a section.

Media playback

Primitive Internet browsers had support for text only, and even the text support was limited to a single font in a single color. Then it appeared browsers with color font and text styles support, as well as pictures. The support for sounds, animations, and videos is handled in different ways by various browsers. Some multimedia elements are supported, and some require an extra helper program, usually called a plug-in. HTML5 represents the standardization of those procedures, because the language itself includes new tags specifically suited for audio and video. Therefore, any browser HTML5-compatible should be able to support them. However, there is not yet an agreement on the codecs that should be supported, and it seems that most of the browsers will wait until the official recommendation in 2014 to take a decision.

As an example, the inclusion of a video in an HTML5 web page can be done as follows:

```html
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
</video>
```

where only the width, height and coded of the video should be provided, as well as the file name and location, while the browser manages the controls and video streaming without the use of plug-ins. This easy implementation and the fact that it
is not necessary to use third-parties software are the main reasons why we have used HTML5 to code the site.

Local Storage

With HTML5, web pages store data locally within the user’s browser. While before, this was done with cookies. HTML5 provides support for Local Storage which is more secure and faster. Data is not included in every server request, but used only when asked for. Moreover, it is also possible to store large amounts of data, without affecting the performance. The data is stored in key/value pairs, and a web page can only access data stored by it. However, the data is stored on client’s hard disk, thus it is not appropriate for our situation since it is foreseen that users access the site by means of multimodal interfaces (e.g. mobile devices or tablets. Note that data cache (data often requested) is solved in this work by using a CDN network, as explained in Chapter 5.

Drag and drop and touch actions

Drag and drop is typically understood in computer GUIs as the action of catching an object and drag it to a different location. This intuitive action has been implemented in HTML5, and it is progressively implemented on modern websites. For instance, the so called Google e-mail manager, Gmail, is able to attach a file by using this action. Other actions adapted to touchscreens are currently being discussed[2].

These features are interesting because they are much appropriated for our site. We think that most of our clients would prefer to use tablets while they are practicing yoga exercises, because these low-weighted devices can be placed anywhere. In this regard, we plan to implement touch actions in our site in the future.

4.2.2 CSS

Cascading Style Sheets (CSS) [3], maintained by the W3C, is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language (e.g. HTML). CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.
4.2 Utilised technologies

For this project, we have customized the CSS styles of our site to adapt to the company colors, since this allows the user to identify the page with that colors and layout and associate them to the company.

4.2.3 Javascript

JavaScript [4] is a scripting language mainly used in web pages. Its syntax is similar to the Java and C languages. Like Java, JavaScript is an object-oriented language.

All modern browsers accept JavaScript code embedded within web pages. This language can be included in any document and is compatible with any Operating System, HTML or whatever runs translated into HTML in the client browser.

Traditionally, it was used in HTML web pages to perform tasks and operations within the client side only, without access to server functions. JavaScript runs on the user agent while the sentences are downloaded with the HTML.

In this work, this technology is utilized for a number of effects and actions. For example, in the homepage we use Javascript to periodically change the presentation images as slides using different effects.

4.2.4 Ajax

Asynchronous JavaScript and XML (Ajax) [5] is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Ajax is not a single technology, but a group of technologies. This combination provides a method for exchanging data asynchronously between browser and server and avoids full page reloads.

We use Ajax to periodically update the chat service without refreshing the full page, which is an important feature in real-time services.

4.2.5 MySQL

MySQL is an open source relational database management system. It is based on the Structured Query Language (SQL) [6], which is used for adding, removing, and modifying information from a database. We will use MySQL to add the users to the database once they are registered, and after that, to validate their data when they
log in.

4.2.6 PHP

PHP [7] is an interpreted programming language originally intended to create dynamic websites. This programming language is mainly used to display dynamic content on websites, so that when performing a request to the server, it executes the PHP scripts associated with that request. The similarity of PHP with C allows programmers to create the most complex applications with a very short learning.

The designed application follows the Model-View-Controller (MVC) architecture [8] (see Fig. 4.7) because it allows intuitively separating the different logical entities that perform each action. The MVC architecture elements functions are next described.

![Model-View-Controller architecture](image)

**Figure 4.7:** Model-View-Controller architecture

**Controller**

The *controller* is the entity who the client can communicate with. A controller is able to use *models* to process the data so as to load *views*. Typically, all PHP web projects use URLs based on the segment-based approach. To exemplify this, consider the following Uniform Resource Locator (URL):

http://tfgcarne.hol.es/yoga/index/home

In this case, we can identify *yoga* as the controller, *index* as the function to be invoked, and *home* as the parameter to be passed to that function. The general form for the URLs is given by:

http://domain/controller/function/parameter
where only the domain segment is mandatory. If no controller is supplied, then the default controller is used (to be configured in section 4.3). If no function is specified then the index function of the corresponding controller (if it is implemented) is called.

Model

The aim of the model is to work with information in a database. For example, a typical model action could be to search (a query) in a database and return the associated data to one controller. Therefore, the controller can make use of this data to load the views.

View

The view is what the client definitively sees at the display. It is simply web code to be processed by the browser. A view can include HTML, Javascript, PHP or any other web programming language understandable by the browser. In general, the controller may load a set of views one after the other, pack them, and send all the code to the client browser in order to elaborate the whole page.

Code Igniter

In order to facilitate the code development, we have used a framework called Code Igniter (CI) [9]. CI is an Application Development Framework - a toolkit - for people building web sites using PHP. Its goal is to enable them to develop projects much faster than writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries. For instance, CI includes a library that search for errors when filling forms, and it is able to detect if a field intended to contain an email address follows the typical structure “(username)@(hostname).(top-level domain)”. CI can be downloaded from [http://ellislab.com/codeigniter](http://ellislab.com/codeigniter), where a functional template can be found. This starting template includes the directory tree used by CI. Fig.4.8 shows the current directory structure of our system, based on the starting template directories.

Next, we detail the content of each directory:

- The application directory contains the models, views and controllers definitions, and also the configuration files.

- Chats contains the logs of the chat room, and also the welcome message to the
non-registered users.

- **Css** contains the style definitions for the views, written in CSS language.
- **Images**, **img** and **media** are devoted to store audiovisual files.
- **Js** contains the Javascript library codes, necessary to implement Javascript methods in our views.
- **Other directories** contain definitions and codes of the CI core and environment and should not be modified.

### 4.2.7 Apache

Apache is the most popular Web server software [1]. It is open source software, which makes it to perfectly fulfill the role of HTTP server and includes the concept of virtual server, responsible for responding to HTTP requests. Apache is very modular. In particular, it is worth pointing out the rewrite module (**mod_rewrite**). This module uses a rule-based rewriting engine, based on a regular-expression parser, to reformulate requested URLs on real-time. Additionally, we use Apache because it is OpenSource, multiplatform, extensible, free and popular, which makes it easy to
obtain and get help.

4.3 Apache and CI configuration

The developed system runs on Apache server with MySQL. Once they are installed, a virtual server has been configured. This is useful to test locally our changes before updating the files on the host server.

The virtual server defines the path where Apache redirects the browser when a localhost request is received. As commented, we will use the URL segment-based approach. However, CI needs that all URLs have the following structure:

\[ \text{hostname/index.php/controller/function/parameter} \]

This is a bit different as compared to the code explained in section 4.2.6 because `index.php` should be added in the beginning of all URLs. We can remove this by using `.htaccess` file and the `mod_rewrite` Apache module. This file is interpreted by the Apache virtual server and can be used to rewrite requested URLs on the fly with some simple rules. Here we show the implemented code, using the \textit{negative} method in which everything is redirected except the specified items:

\begin{verbatim}
RewriteEngine on
RewriteCond $1 !^ (index.php|css|js|images|img|media|chats)
RewriteRule ^(.*)$ index.php/$1 [L]
\end{verbatim}

In the above code, any HTTP request other than those for \{index.php, css, js, images, img, media, chats\} is treated as a request for the `index.php` file, thus it is rewritten. On the other hand, the [L] flag causes \textit{mod_rewrite} to stop processing the next rules.

Next, we will explain the different configuration files that have been modified with respect to original framework package:

- `autoload.php`: this file contains the helpers and libraries to be loaded automatically. Although it can be done at the beginning of each controller, it is more suitable to define them here. We modify the following lines:

\begin{verbatim}
$autoload['libraries'] = array('session', 'database');
$autoload['helper'] = array('url', 'form', 'file');
\end{verbatim}
4.4 Web implementation

- config.php: this file contains general CI configuration parameters. The following line has been modified:

```php
$config['javascript_location'] = '/js/jquery-1.5.1.min.js';
```

in order to tell CI where the javascript sources can be found.

- database.php: here we can set the database parameters, such as the hostname, username, password, database name and others. For example:

```php
$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'annacarne';
$db['default']['password'] = 'telematic';
$db['default']['database'] = 'test_database';
$db['default']['dbdriver'] = 'mysql';
```

- routes.php: this files can be used to re-map Uniform Resource Identifier (URI) requests to specific controller functions. We have used that to set the default controller through the following line:

```php
$route['default_controller'] = "yoga";
```

4.4 Web implementation

In this project, we have implemented a number of controllers, models and views to accomplish the desired functionalities. This section describes all these entities, summarized in Fig.4.9. Three controllers have been implemented: Yoga, Chat and Register. The controller Yoga is devoted to load views, and will be next explained, while controllers Chat and Register will be addressed in sections 4.4.1 and 4.4.2. We define only the model user_model, explained in section 4.4.1 when describing the registration implementation. And finally, several views are defined; some of them will be explained in this section and the remaining in 4.4.1 or 4.4.2, depending on their objective.
We start this section describing the controller Yoga. This controller has been implemented as follows:

```php
<?php
class Yoga extends CI_Controller {
    function __construct() {
        parent::__construct();
        /*
        * $this->load->library('session');
        * $this->load->helper('url');
        * $this->load->helper('form');*/
    }
    function index($param='home') {
        $this -> load -> view('head');
        $this -> load -> view('header_menu');
        $this -> load -> view($param);
        $this -> load -> view('footer');
    }
}}?>
```
The first part of the code corresponds to the controller constructor, where typically additional libraries and helpers could be loaded. However, since we have configured these files to be autoloaded (see previous section), this is not necessary and it is shown as commented. This controller has simply one function, index, whose aim is to load the desired view, specified by param, as well as other necessary views at the same time. If no parameter is passed, therefore the view home is loaded, set as the default value for param. Finally, notice that according to all parameters configured and how this function is implemented the following 3 URLs

http://annacarnevila.proves.org/
http://annacarnevila.proves.org/yoga/index/
http://annacarnevila.proves.org/yoga/index/home

result in the same output. The view home is the homepage of the site, shown in Fig.4.1 (section 4.1). In this view, we show images as a presentation, what is called a slider, with the aim of presenting the company to a new user. On the other hand, some site news and comments are shown intended to common visitors.

All pages have a specific structure, as can be seen in the figures showing the site in section 4.1. Specifically, some parts of the page are maintained and are common to all the pages. The views that contain this common code to create all pages are remarked in Fig.4.9, and they are head, header_menu and footer. Here we describe the main functionality of each of these views, and the corresponding code is shown in Appendix A:

- head contains all the code necessary to correctly include all Javascript and CSS files that will be used to give effects and format to the page
- header_menu describes the navigation bar included in all pages. In addition to the links among pages, this bar contains the login form, where users can introduce their username and password in order to get permission to, for instance, participate in the chat room.
- footer implements the part at the bottom of the page, containing the twitter widget, some navigation links, the search bar, the authors information and other stuff.

### 4.4.1 Registration implementation

Any business aiming to sell products via the Internet needs a registration to gather the client’s information. The need for this module is twofold: to differentiate content
and functions from users, and manage the user data.

On the first place, a username and password for each client allow the business owner to show different content depending on who is accessing to the web page. In our case, each user should be able to watch only the videos corresponding to those courses previously purchased. Or, for instance, we can show content in one language or other, depending on which user is logged in to the site. Additionally, we can set some functions to be only accessible for registered users. We implement this last function for the chat, where only registered users can participate.

On the other hand, storing the billing data makes easy the client to purchase new courses, and alleviates the task of introducing each time his/her information. This is important because often some clients decide at the last step to cancel the process, because of the laziness of introducing that information. It is therefore, necessary to make things easy from the point of view of the clients. However, since we have not implemented the payment platform yet, this function is currently not active. In any case, we think that it would be easy to add it to the current registered users by adding some fields to existing database entries.

The registry is managed through different entities: the controller \texttt{register}, which manages login, registration and logout of users, and the \texttt{user\_model}, who access to the user database and check or add data. In addition to this, there are two views related with the registry functionality: \texttt{register\_view} and \texttt{thank\_view}, as well as the login/logout box of the \texttt{header\_menu} view which is embedded in all pages.

At this stage of the project, the register of users provides three basic functions for our web application: log in, log out and sign in, explained next.

\textbf{Log in}

Fig.4.10, shows a diagram describing this functionality. Assume that we have a registered user. Then, in any of the pages we can fill in the form in the \texttt{header\_menu} view and click on the button Enter. This action sends a POST request to the server to the function \texttt{login} of the controller \texttt{register}. This function is shown below:
4.4 Web implementation

```php
public function login() {
    $email = this -> input -> post('email');
    $password = this -> input -> post('pass');
    if (this -> user_model -> login($email, $password))
        redirect('yoga/index/sala_chat');
}
```

**Figure 4.10:** Log in functionality

First, data is retrieved by using the CI helper input and the function post. Therefore, they are and passed to the model user_model. This entity is the responsible of communicating with the database through the login function, to verify the client credentials and start a new session if needed. The function `login` of the model `user_model` is next shown:
4.4 Web implementation

```php
function login($email, $password) {
    $this->db->where("email", $email);
    $this->db->where("password", $password);
    $query = $this->db->get("user");
    if ($query->num_rows() > 0) {
        foreach ($query->result() as $rows) {
            $newdata = array(
                'id' => $rows->id,
                'name' => $rows->username,
                'email' => $rows->email,
                'logged_in' => TRUE,
            );
        }
        $this->session->set_userdata($newdata);
        return true;
    }
    return false;
}
```

In this function, we first create the rules to search on the database. Therefore, the table user of the database is analyzed, creating an array with the corresponding user information. Finally, a new session is started with the obtained data.

Back to the function login of the controller, it finally redirects the browser to the `sala_chat` view, where the client will be able to participate in the chat room if it has started its session successfully.

**Log out**

A similar diagram is shown for this function in Fig. 4.11. In the diagram, we set as an example the initial point at the chat room, although the process may be started from any of the pages (as long as a previous session has been started) when the user clicks on the button Log out from the view header_menu. This button redirects to the function logout of the controller register, whose aim is to destroy the session and redirects to the homepage. In order to avoid redundancy, these codes are deferred to Appendix A.
Sign in

Finally, the diagram describing how new users are added to the registry is depicted in Fig. 4.12. As other functions, from any of the pages the action starts when a non-logged in user clicks on the button *No estas registrado?*. This calls the function `signin` of the controller `register`, which redirects to the registration page by means of the controller `yoga`. The data in the form is sent as a POST request and processed by function `registration`, which validates the data using a CI helper as shown below:
4.4 Web implementation

No estás registrado?

C: Yoga
F: Index
P: home.php

Proceder

C: Register
F: signin
redirect
/yoga/index/register_view.php

Gracias

Volver a la página de inicio

C: Register
F: registration
retrieve and validate data
M: USER_MODEL          add user         DB

Figure 4.12: Sign in functionality
public function registration() {
    $this->load->library('form_validation');
    // field name, error message, validation rules
    $this->form_validation->set_rules('user_name', 'UserName', 'trim|required|min_length[4]|xss_clean');
    $this->form_validation->set_rules('email_address', 'YourEmail', 'trim|required|valid_email');
    $this->form_validation->set_rules('password', 'Password', 'trim|required|min_length[4]|max_length[32]');
    $this->form_validation->set_rules('con_password', 'PasswordConfirmation', 'trim|required|matches[password]');

    if ($this->form_validation->run() == FALSE) {
        $this->signin();
    } else {
        if ($this->user_model->add_user()) $this->thank();
        else $this->signin();
    }
}

Therefore, this function communicates with the function add_user of the model user_model in order to add a new entry to the database. The last action of this function is to show a view in order to thank the client, loaded by means of controller yoga. Finally, this page contains a button that redirects to the homepage, where the client may login using the information of the new registered user.

4.4.2 Chat implementation

The added value of this project starting with this work with respect to other e-learning projects is that gives the possibility to carry on chat or videoconferences among teacher and students. This functionality is very important because some students will leave the courses when they encounter with the first problems. In this context, to have a few words with the teacher can be beneficial to motivate again the student and reenlist again to the courses. Even more in this case, where what is learnt is a physical activity, maintaining a videoconference with the teacher can be helpful for the student to resolve some doubts and correct some bad practices. In this project, due to limited time, only the chat has been implemented. This is the aim
of this section: to present the fundamental concepts regarding the chat functionality implementation. To begin with, there are two possibilities for the chat functionality: let any visitor to write a message, or give permission only to registered/selected users. The implemented chat follows the second approach. Therefore, visitors are not able to neither watch the previous chats nor participate in them unless they sign and log in. Additionally, it is almost mandatory that the chat is automatically updated without refreshing the current page in the browser. We implemented this by using Ajax, because it allows sending data to, and retrieving data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Two entities are the basis of the chat: the controller chat and the view `sala_chat`. By the moment, there are only two conversations, saved to the directory chats at the root directory, i.e. /chats. As commented in section 3, in future we plan to manage and save the conversations by means of the database, thus it is a temporary solution. They are log and welcome chat. The log file contains the conversations maintained using the chat room and it is modified each time a registered user sends a message. The welcome chat is what the non-registered users watch at the chatroom, and it is static. The chatbox is located at the view sala_chat, whose content depends on the session information. This is determined when the page is charged, as follows:

```php
<?php
  $name = $this -> session -> userdata('name');
  if ( (!isset($name) || empty($name)))
    echo read_file('./chats/welcome_chat.html');
  else
    echo read_file('./chats/log.html');
?>
</div>
```
```html
var usermsg = document.getElementById('usermsg');
<?php $name = $this -> session -> userdata('name');
  if ( (!isset($name) || empty($name)))
    echo 'usermsg.disabled=true;'
?>
```
In addition to this, we show the Javascript code that disables the usermsg field. This is important because we want non-registered users neither to watch the previous chats logs nor leave comments. Fig. 4.13 shows a diagram explaining how the chat works when a message is introduced. All actions need to pass through the chat controller. When a user enters a message and press the key Enter, some Javascript code (see Appendix A for a detailed explanation) takes this input and sends a POST request to the function agregar of the controller chat. This function manipulates the message to apply a format. For example, if the user AnnaCarne enters the message “Hello”, this function redefines the message as “(26/6/13 20:51) AnnaCarne: Hello”. This can be seen next, where we show how the PHP function date can be used to retrieve the current date and hour. Similarly, the username and message can be retrieved by using the CI helpers session and input:

```php
function agregar() {
    $mensaje = "
    <div class='msgln'>
    <p> ( " . date("j/n/y G:i") . " ) <b>" .
        $this -> session -> userdata('name') . "</b>:" .
        $this -> input -> post('mensaje') . "
    </p><br>
    </div>";

    if (!write_file('./chats/log.html', $mensaje, 'a'))
        echo 'Unable to write the file';
    else echo 'File written!';
}
```

Finally, this function appends to the file log.html the formatted message.

Back to the Javascript code in the view sala_chat, after sending the message to the function agregar it loads the function loadLog. This function is executed periodically and also once after sending each message. The aim of this function is to update the chat box. In this regard, this function uses Ajax to send requests to the function actualizar of the controller chat, whose code is detailed next:
Figure 4.13: Chat functionality

```php
function actualizar() {
    $name = $this->session->userdata('name');
    if (!isset($name) || empty($name))
        echo read_file('./chats/welcome_chat.html');
    else
        echo read_file('./chats/log.html');
}
```

This is similar to what is done for the chatbox when it is loaded.
As the reader can see from loadLog code, the function *actualizar* simply returns the content of the chat log. It is contained on the parameter *msg* and is used to update the chatbox. Finally, in order to give advice the user that a new message is present at the chatbox, the scroll of the element is modified by means of the Javascript function *animate*. Note that we set a high number to overpass the size of the element and therefore reach the bottom of it.
Chapter 5

Content Delivery Network

In this chapter, we show how the Web was modified so as to use a CDN to address the incoming requests. First, the concept of CDN will be introduced, with special attention to its functionalities, interactions with existing entities, requirements and benefits. Next, section 5.2 addresses an introduction to Coral CDN, the CDN service that has been used. After this, we describe the main protocols used. This can be useful to the reader so as to better understand section 5.4, where we show the simulations with and without Coral CDN, and draw some conclusions based on the obtained results.

5.1 Introduction

CDN [10, 11, 12, 13, 14], was developed in 1998, replicate contents over several mirrored web servers (i.e., surrogate servers) strategically placed at various locations in order to deal with the flash crowds. Geographically distributed web servers’ is commonly used by service providers to improve performance and scalability. A CDN has some combination of a content-delivery infrastructure, a request routing, distribution and accounting infrastructure. CDNs improve network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication and thus offer fast and reliable applications and services by distributing content to cache servers located close to users. In Fig.5.1, we can see the comparison of a current network with and without CDN architecture, whereas Fig. 5.2 provides the high level view of the basic interaction flows among the components in a CDN environment. The interaction flows are the following:
5.1 Introduction

Figure 5.1: Typical network vs CDN

Figure 5.2: Basic interaction flows in a CDN environment
5.1 Introduction

1. The customer requests content from the origin server by specifying its URL in the Web browser.

2. When the origin server receives a request, it makes a decision to provide only the basic contents, e.g. index page of the site.

3. To serve the high bandwidth demanding and frequently asked contents (e.g. embedded objects-fresh content, navigation bar, banner ads etc.), the origin server redirects client’s request to the CDN provider.

4. Using a selection algorithm, the CDN provider selects the surrogate server which is closest to the client, in order to serve the requested embedded objects.

5. Selected surrogate server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

The main functionalities and services of a CDN network are the following:

- storage and management of content
- distribution of content among surrogates
- cache management
- delivery of static, dynamic and streaming content
- backup and disaster recovery solutions
- monitoring, performance measurement and reporting

As we explained before, using a CDN network has a wide number of benefits. The customer needs to invest in Web site infrastructure is reduced, as well as the operational costs of managing such infrastructure. Moreover, the content delivery quality, speed and reliability are improved. From the operator point of view, the traffic jams are bypassed on the Web, since data is closer to user and there is no need to traverse all of the congested pipes and peering points. Finally, a customer that makes use of a CDN network is interesting for the hosting company, because the load on origin servers is also reduced. Finally, using a CDN network can help to the Search Engine Optimization (SEO) of the site. For all these reasons we consider to use a CDN network in our Web server, where we want to make our site able to stream audiovisual contents (basically videos) simultaneously to the maximum number of users.
5.2 Coral CDN

The specific CDN network that has been used for this work is CoralCDN \cite{15, 16}. In this section, we introduce its origins, how it works and how we use this CDN to deliver our site content.

Introduction

CoralCDN is an open-source content distribution network based around peer-to-peer technologies, comprised of a world-wide network of web proxies and name servers. We have chosen CoralCDN because, at this stage, the aim of the project is to show a proof of concept. In other words, we will use this free CDN implementation to simulate the environment and get insight into the behavior of the system with and without the CDN network.

Publishing through CoralCDN is as simple as making a small change to a URL’s hostname; a decentralized DNS layer transparently directs browsers to nearby participating cache nodes, which in turn cooperate to minimize load on the origin webserver. CoralCDN proxies automatically replicate content as a side effect of users accessing it, improving its availability. Using modern peer-to-peer indexing techniques, CoralCDN will efficiently find a cached object if it exists anywhere in the network, requiring that it uses the origin server only to initially fetch the object once.

The goal of CoralCDN was to make desired web content available to everybody, regardless of the publisher’s own resources or dedicated hosting services. To do so, CoralCDN provides an open, self-organizing web CDN that any publisher is free to use, without any prior registration or authorization. Publishing through CoralCDN is as simple as appending a suffix to a URL’s hostname, e.g., \texttt{http://example.com.nyud.net/}. This URL modification may be done by clients, origin servers, or third parties that link to these domains. This approach is a bit different than typical CDN networks, where the origin server redirects the user’s request to the CDN network. In this case, clients access to such “Coralized” URLs, and they are transparently directed by CoralCDN’s network of DNS servers to nearby participating proxies. These proxies, in turn, coordinate to serve content and thus minimize load on origin servers. This is not so much practical, because the client has to access to the “Coralized” URL in order to be served from the CDN network. As we will see next, we will implement a rewriting method at the server side so as to rewrite the URLs.
System overview

At a high level, the following steps occur when a client issues a request to CoralCDN, as shown in Fig. 5.3:

1. Resolving DNS. A client resolves a “Coralized” domain name (e.g., of the form example.com.nyud.net) using CoralCDN nameservers. A CoralCDN nameserver probes the client to determine its round-trip-time and uses this information to determine appropriate nameservers and proxies to return.

2. Processing HTTP client requests. The client sends an HTTP request for a Coralized URL to one of the returned proxies. If the proxy is caching the web object locally, it returns the object and the client is finished. Otherwise, the proxy attempts to find the object on another CoralCDN proxy.

3. Discovering cooperative-cached content. The proxy looks up the objects URL in the Coral indexing layer.

4. Retrieving content. If Coral returns the address of a node caching the object, the proxy fetches the object from this node. Otherwise, the proxy downloads the object from the origin server example.com.

5. Serving content to clients. The proxy stores the web object to disk and returns it to the client browser.

Figure 5.3: Steps to serve a Coralized URL
6. Announcing cached content. The proxy stores a reference to itself in Coral, recording the fact that is now caching the URL.

Implementation

Coral CDN can be implemented in Apache servers by means of using .htaccess file, similarly to the one we used to avoid writing `index.php` in all our URLs because of using CI. We need to write `nyud.net` at the end of the domain name of our URLs. This allows redirecting all the HTTP requests to the Coral CDN servers. We have implemented this by using the following code:

```
RewriteCond %{HTTP_USER_AGENT} !* CoralWebPrx (1)
RewriteCond %{QUERY_STRING} !(\^|&) coral-no-serve$ (2)
RewriteRule ^(.*)$ http://tfgcarne.com.nyud.net/$1 [R, L] (3)
```

where the *negative* method is used. As a consequence, everything is redirected except the specified items. The syntax of this file makes use of regular expressions, but since they are out of the scope of this work, we simply explain the intuitive meaning of each line:

1. (1) ensures that the request is not coming from Coral in order to prevent an infinite loop.

2. (2) explores the URL to search for “?coral-no-serve” or “&cora-no-serve”. In such a case, this means that there is an error with coral and this URL cannot be coralized.

3. (3) does the actual URL rewriting. If the previous two conditions are satisfied (the request is not from Coral itself, nor it is a failed Coral request), then do the rewrite. In this case, we append “nyud.net” to our domain name in order to get coralized URL. The flags [R] and [L] stand for redirect and last. Redirect causes the browser to redirect to the URL, while last means that there are no more rewriting rules to be processed. As the reader can notice, if we use Coral CDN we have to remove the [L] flag from the code used to write “index.php” in the specified URLs.
5.3 Protocols used

This section gives an overview of the main protocols used in this work, for an easier understanding of the simulations in next section:

**IP**

The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite, also known as TCP/IP, for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source to the destination hosts solely based on the IP addresses. For this purpose, IP defines datagram structures that encapsulate the data to be delivered, as well as addressing methods used to label the datagram with source and destination information.

**TCP/UDP**

Transmission Control Protocol (TCP) [17] is a transport layer protocol that provides reliable, ordered, error-checked delivery of a stream between programs running on computers connected to a local area network, intranet or the public Internet. Web browsers use TCP when they connect to servers on the WWW, and it is used to deliver e-mails and transfer files from one location to another.

Applications that do not require the reliability of a TCP connection may instead use the connectionless User Datagram Protocol (UDP), which emphasizes low-overhead operation and reduced latency rather than error checking and delivery validation.

In our case, UDP can be used on the connection among the closer surrogate server and the clients (see Fig.3.1 and 3.2), in order to increase the throughput. Nevertheless, this protocol cannot be used on the other connections since UDP requests cannot travel through firewalls. Hence, for the other connections TCP should be used.

**HTTP/RTSP**

The Hypertext Transfer Protocol (HTTP) [18] is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.
Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

HTTP acts as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, returns a response message to the client. The response contains completion status information about the request and may also contain requested content.

HTTP uses different request methods as we will see in the simulations. The more relevant and useful for us are

- Get: Requests data from a specified resource.
- Post: Submits data to be processed to a specified resource. It is often used when uploading a file or submitting a completed web form.

The Real Time Streaming Protocol (RTSP) is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Clients of media servers issue VCR-like commands, such as play and pause, to facilitate real-time control of playback of media files from the server. The transmission of streaming data itself is not a task of the RTSP protocol.

HTTP Streaming Protocol Extensions or HTTP progressive download from a web server are useful in case of streaming pre-encoded content and it is not relevant the end-to-end delay. On the other hand, if the content is a live broadcast or it is necessary to reduce the end-to-end delay, then RTSP might be the best choice. In our case, HTTP is enough in order to serve the courses, since it is not necessary that they are real-time watched. However, if we expect to implement a videoconference service, maybe we would have to consider RTSP.

5.4 Simulations

We have captured the traffic in our network by means of Wireshark, an open-source IP packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. With this application we are going to analyze bandwidth and IPs of the streaming download process with and without using the CDN.
5.4 Simulations

Non-CDN simulations

We show how the client sends an HTTP GET packet in order to retrieve the video. After that the server answers this request delivering the requested video by means of TCP streams. This can be seen in Fig. 5.4: As we can see the client with the IP: 192.168.1.10 asks to the server with the IP: 209.190.85.4. Similarly, the same information can be shown as a flow graph as depicted in Fig. 5.5.

Moreover, in that figure we can see the TCP traffic. This traffic consists on the video streams that are exchanged among the server and the client, and the corresponding acknowledgments.
5.4 Simulations

Now, we are going to get all the video streams and filter all other packets. Wireshark provides a tool called IO Graph able to print the download speed in bits/tick where each tick can be scaled in seconds. By setting 1 tick = 1 sec, we obtain the results shown in Fig. 5.6. There we show the graph, as well as the bitrate of the video, obtained by using the program ffmpeg available at any UNIX environment. Therefore, by using all the information shown in the Fig. 5.6 we can predict how many users will be able to play the video at the same time, as follows:

\[
\frac{BW}{N_{\text{users}}} = CBR
\]  

(5.1)

where \( CBR \), \( BW \) and \( N_{\text{users}} \) stand for the video Constant Bit Rate (CBR), the Bandwidth (BW), i.e. the server downlink throughput, and the numbers of users playing video simultaneously, respectively.

![Figure 5.6: IO Graph](image)

In this case, \( BW = 16 \text{Mbits/s} \) and \( CBR = 613 \text{kbits/s} \). Hence, by using (5.1), we obtain that at most 32 users can play the video simultaneously. Notice that although we have assumed a very simplified model in (5.1), it is realistic to assume that the bandwidth decreases as the number of users increases. To verify this, we capture the traffic in the network when the same video has been simultaneously delivered to 3
users, obtaining the results depicted in Fig. 5.7. In this case, each user has been served at a mean throughput of 2\text{Mbit/s}, being a lower value as compared to the single-user case.
CDN simulations

Now, we aim to capture the traffic when using the same services as previous section but using CoralCDN. First, we can see in Fig.5.8 how the client sends an HTTP GET request to the server. In this case, we can see that the URI has been rewritten in order to redirect the request to the CDN network. For this reason, in the received packet we have different IPs.

![Figure 5.8: HTTP GET video request using CoralCDN](image-url)

In particular, we can see that the Server IP address is **195.113.161.83**. However, if we follow the TCP stream (shown in the Fig.5.9), we obtain that there are two different IP address. They correspond to the proxy who redirects the request and the closer server. Additionally, as in previous section, we can see the flow graph in Fig.5.10.

Finally, notice that we cannot calculate the maximum number of users as in (5.1) because in this case the number of servers is higher than one. In fact, the equivalent bandwidth can be assumed arbitrarily high and therefore, ideally, there is no limit in the number of users to be served.
5.5 Conclusions

We have captured the traffic when the server uses the CDN network and when it works by itself. Results show that the hosting server cannot serve more than 30 users approximately at the same time. Therefore, assuming that each class is watched by a minimum of 4 students at the same time, we can estimate that the maximum number of videos that can be deployed is 7.

Our yoga company partner plans to offer a minimum of 8 courses, where each course consists of at least three videos. Moreover, at the present time, 4 classes on average are provided every day. He is planning to record them (maybe not all of them but 1 for the morning and 1 for the afternoon), and upload the videos to the site. Consequently, it seems that it is really necessary to use a CDN network because one server only is not able to cater the expected traffic.
Chapter 6

Budget

In order to develop the budget properly, we will split this section in four fundamental parts: Development, Test, Maintenance, and Enhancement phases.

To make the project feasible, an initial investment of 50,000€ is necessary. In Appendix B, we show the Profit and Loss (PL), Cash Flow Statement (CFS) and balance sheet for each year. Therefore, we will see that after three years we obtain a 15% of Return on equity (ROE).

6.1 Development phase

To begin with the project, the student will need some time to learn the technologies to be used, design and implement the system, and make some functional tests. We estimate that the duration for the development phase will be about 5 months.

Given the different sides that the system presents, we are going to need technical staff, able to deal with and handle the tools needed to solve it. Given that the development will be quite intensive, we plan a tough timeline enough to finish the project, thus the maximum number of hours that can be spent with a high level performance. We estimate that 6 hours a day can achieve results without burning the employee. Moreover, the type of work gives the flexibility of being able to work at home.

As for the salary, since we are dealing with a qualified engineer with an intensive workload, we think 25€ to be a fair hourly wage. Therefore, considering 20 days a month of work, the sum salary per month results on 3,000€. Adding the Social Security plus and other taxes, which can be approximated as a 30% of those 3,000€, the final employee cost per month is around 3,000+900=3,900€.
6.2 Test phase

Regarding to the hardware, a personal computer will be necessary, for which 1.200€ will be allocated. We assume that the amortization cost for the computer is linear. Considering that the computer can be used 5 years, and after this period its value is equal to 300€, each month the amortization cost is going to be equal to 15€.

On the other hand, all the necessary software to develop this project phase is free, except the domain, which costs 20€ per year. For the next phases, we are going to rent paid services, since we need more functionalities and confidence that cannot be ensured by using free software services.

Finally, it is necessary to rent a place where the project is carried out. In this case, a furnished office will be rented, and we assume an approximate cost of 250€ per month. Moreover, taking into the typical bills, such as water, electricity and ADSL connection, we need to add an additional per month cost of 100€.

6.2 Test phase

In order to make more robust the system and promote the business, 7 months will be offered as free to the clients. This phase will be useful to detect bugs from unexpected actions of users and perform load analysis, as well as retaining our first clients.

For this phase, since the system will be already designed and the task is easier, one engineer on part-time regime will be hired (if possible the same person) with a monthly salary of 2.000€ plus taxes.

In order to develop this phase, some software services will be acquired. We need an Secure Socket Layer (SSL) certificate, a hosting service, and a CDN service. For the SSL certificate, we estimate a related cost of 200€ per year. On the other hand, the hosting service will have the following capabilities:

1. Disk Storage: Unlimited\(^1\)
2. Full Shell/SSH/FTP/SFTP Users: Unlimited\(^1\)
3. MySQL Databases: Unlimited\(^1\)
4. SSL Secure Server (request unique IP)
5. Enhanced Web Security
6. WebDAV: a file manager especially suited for web servers

---

\(^1\)Unlimited resources are not necessary, but most of the servers provide them.
6.2 Test phase

7. Daily Access Statistics
8. Access to raw log files
9. Directories with password
10. .htaccess capability
11. PHP5 Support
12. Crontab Access, an application to schedule some periodic tasks
13. Subversion Repository

A server like this may cost approximately 8€ per month.

Regarding to the CDN service, it should be able to afford the expected demand of high quality audiovisual contents streamed to many users at the same time. For the proper estimation of the CDN service cost, let us assume that for every 100 clients that access to the site, only 40 of them watch some course. In particular, we consider that these 40 users are distributed as follows:

- 5 of them only watch one video
- 20 of them watch from 2 to 5 videos
- 15 of them watch from 6 to 10 videos

Consequently, the average number of videos watched out of 100 visitors can be computed as follows:

\[
\left( 5 \cdot 1 + 20 \cdot \frac{2 + 5}{2} + 15 \cdot \frac{6 + 10}{2} \right) = 195 \text{ videos} / 100 \text{ visitors}
\] (6.1)

Therefore, assuming that, thanks to the marketing, a minimum of 600 clients access to the page each month, we have that 1.170 videos will be requested. Additionally, let us suppose that each video will be seen 5 times on average. As a provider, we will use Amazon CloudFront [19], because of its popularity and good quality-pricing tradeoff. The cost for each HTTP request using this provider is equal to 0,0068€, thus we end up with a total amount of 39,78€ per month during this phase.

Finally, we will need a way to give out our product on the Internet. Our marketing will be based on Google Adwords campaigns [20], since it is a powerful tool that allows anyone with a small budget to appear in Google searches. The major feature is that

\footnote{This will be useful for security reasons, as well to test new features}
6.3 Maintenance phase

it is paid per clicks, i.e. only each time someone clicks to the ad, but not when it appears. The cost for each click is related to relevance of the keyword that originates the search. In our case, we will use yoga as keyword. This is not a common or popular keyword, so we estimate the average cost per click for this word as 0.25€ per click. Consequently, for each 100 clicks this supposes 25€. At the beginning, almost all the people will know about the site existence by means of Google ads. Therefore, to afford the 600 monthly visitors considered, we need to invest approximately 150€ per month.

Another way we will promote our business is by giving handouts at the beginning and the end of in-person classes. The total amount devoted to this issue will be 10€.

As can be seen in Appendix B, after the first year 43.754€ are spent, thus the costs cover almost all the investment during this year.

6.3 Maintenance phase

After testing the system with users, we plan to spent 1 year maintaining the site, as a payback period. Consequently, our clients should pay to watch the courses and lessons, thus we elucidate the earnings and costs expected during this phase.

First, we will maintain the same staff for this phase. The wage will be increased according to the Índice de Precios al Consumo (IPC) variation at the beginning of each year, resulting on 2.050€ per month plus taxes. The taks for this employee will be:

- Maintain and update the site
- Take over the marketing: SEO, Google Adwords, handouts, social networks, etc.

Regarding to the technical resources, out of those 1.170 videos requested, a 20% of them will be free offered. Therefore, since each purchased video generates an HTTPS request, we can approximately estimate it to 936 HTTPS requests per month. The cost of each HTTPS request is 0.0092€, thus it results 8.61€, to be added to the HTTP requests when each client watch the video. In order to be conservative, we will consider a 30% of deviation for these results, so overall we end up with 62.90€ per month devoted to the CDN costs.

In what follows, we will estimate our earnings by considering the same distribution commented before when calculating CDN costs. Since we plan to set the average cost
6.4 Enhancement phase

per video to 3.50€, we will earn approximately $195 \cdot 3.50 = 682.50€$ out of 100 clicks (see (6.1)). Consequently, we obtain $682.50 \cdot 600/100 = 4.095€$ per month by means of the promotion on Google Adwords. Additionally, we expect that the revenues per month can be increased by 10% due to the handout investment. Moreover, we have taken into account the initial state where most of the people that purchase videos are new clients. As long as the site becomes more popular, we expect that some of the clients will become more habitual, such that some months our earnings can increase. In this regard, we will apply a different factor for each month, denoted as the profit rate, able to capture these possible variations. This magnitude will be estimated based on the yoga company partner experience of clients’ trends. For example, it is known that the demand on months corresponding to summer is lower than for the rest of the year.

We expect to obtain a positive PL for this year, equal to 19.222€. Consequently, during this phase we start to compensate the initial investment.

6.4 Enhancement phase

The last phase considered in this work is the enhancement phase, carried out during the third year and beyond.

During this phase, we expect to translate our site to other languages, in order to extend the business to other countries where yoga is also or even more popular, e.g. United States. In this regard, we will hire a person to translate the site and subtitle the videos. It can be an university student with a trainee contract working on part-time regime, and a monthly salary of 650€ plus taxes.

To develop his/her work, we will buy another laptop, with a value of 800€, and amortization calculated similarly to the other laptop.

We expect that by translating the content, the average number of visitors grows from 600 to 800 per month, thus we recalculate the marketing, CDN costs, and also the expected earnings. Results can be found on Appendix B, third year. To summarize, we expect that after this third year, we can recoup the full initial investment and bring the first benefits.
Chapter 7

Conclusions

Starting from the willing of a yoga company owner of extending his business over the Internet has lead to the idea of designing and develop a platform able to provide audiovisual teleducation services. The interest of this project for a Yoga company is twofold: to give service at anywhere and anytime and to use the web to attract new clients.

To this end, the project has focused on two different aspects: (1) provide an added value with respect to similar business already deployed, and (2) make the system scalable with the number of users.

With respect to (1), we have endowed our service with the added value of maintaining real-time conferences among the teacher and students. This is important to resolve doubts and motivate the students during the learning process.

On the other hand, to manage the expected traffic flows, (2), it is necessary to use a decentralized architecture with the use of surrogate servers as described in Chapter 3. To do so, we have used a CDN service, able to redirect the traffic closer to the user. Consequently, content delivery quality, speed and reliability may be improved.

Additionally, it has been developed a budget taking into account marketing costs and over-the-year earnings variations according to our partner experience. An important reference for the student has been the assistance to the free UPC seminar: “Loop! Marketing Online de mejora continua”, impart by Toni Padrell. The analysis validates the feasibility of the project, and shows the return of investment after the first 3 years.

A number of future lines after this work remain open. We have enumerated some of them throughout this work, being summarized next:
• Extend the chat functionality to a videoconference service.

• Adapt the data asked to the students during the registry, as well as validation of users. For instance, sending an e-mail to validate the account or using a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [21] tests. Moreover, we plan to implement different user profiles, where each profile gives some advantages based on the price. For example, the Premium profile could be the one that can watch the videos a non-limited number of times.

• Implement a shop cart and payment service using SSL procedures for secure transactions.

• Analyze the current CDN and hosting providers, according to what is needed. As result, hiring appropriate CDN and hosting services.

• Make use of Google Adwords or other marketing services (SEO) in order to make known the business and attracting clients.
Appendix A

Code

A.1 Controllers
A.1.1 Chat

```php
<?php class Chat extends CI_Controller {
    function __construct() {
        parent::__construct();
    }
    function agregar() {
        $mensaje = "
            <div class='msgln'>
            <p> (" . date("j/n/y G:i") . ") </p>
            $this -> session -> userdata('name') . 
            $this -> input -> post('mensaje') . "
            </div>
        <br>
        <div>
            if (!write_file('./chats/log.html', $mensaje, 'a'))
                echo 'Unable to write the file';
            else echo 'File written!';
        }
    function actualizar() {
        $name = $this -> session -> userdata('name');
        if (!isset($name) || empty($name))
            echo read_file('./chats/welcome_chat.html');
        else
            echo read_file('./chats/log.html');
    }
}?>
```
A.1 Controllers

A.1.2 Register

```php
<?php class Register extends CI_Controller {

    function __construct() {
        parent::__construct();
        $this->load->model('user_model');
    }

    function signin() {
        redirect('yoga/index/register_view');
    }

    function login() {
        $email = $this->input->post('email');
        $password = $this->input->post('pass');
        if ($this->user_model->login($email,$password))
            redirect('yoga/index/sala_chat');
    }

    function thank($param=' ') {
        redirect('yoga/index/thank_view');
    }

    function registration() {
        $this->load->library('form_validation');
        // field name, error message, validation rules
        $this->form_validation->set_rules('user_name', 'UserName', 'trim|required|min_length[4]|xss_clean');
        $this->form_validation->set_rules('email_address', 'YourEmail', 'trim|required|valid_email');
        $this->form_validation->set_rules('password', 'Password', 'trim|required|min_length[4]|max_length[32]');
        $this->form_validation->set_rules('con_password', 'PasswordConfirmation', 'trim|required|matches[password]');

        if ($this->form_validation->run() == FALSE) {
            $this->signin();
        } else {
            if($this->user_model->add_user()) $this->thank();
            else $this->signin();
        }
    }

}?
```
function logout() {
    $newdata = array(
        'user_id' => '',
        'user_name' => '',
        'user_email' => '',
        'logged_in' => FALSE,
    );
    $this->session->unset_userdata($newdata);
    $this->session->sess_destroy();
    redirect('');
}

A.1.3 Yoga

<?php class Yoga extends CI_Controller {
    function __construct() {
        parent::__construct();
        /*$this->load->library('session');
        $this->load->helper('url');
        $this->load->helper('form');*/
    }
    function index($param='home') {
        $this->load->view('head');
        $this->load->view('header_menu');
        $this->load->view($param);
        $this->load->view('footer');
    }
}?>
A.2 User model

```php
<?php
class User_model extends CI_Model {
    function __construct() {
        parent::__construct();
    }

    function login($email, $password) {
        $this->db->where("email", $email);
        $this->db->where("password", $password);
        $query = $this->db->get("user");
        if ($query->num_rows() <= 0) return false;
        foreach ($query->result() as $rows) {
            $newdata = array(
                'id' => $rows->id,
                'name' => $rows->username,
                'email' => $rows->email,
                'logged_in' => TRUE,
            );
        }
        $this->session->set_userdata($newdata);
        return true;
    }

    function add_user() {
        $data = array(
            'username' => $this->input->post('user_name'),
            'email' => $this->input->post('email_address'),
            'password' => md5($this->input->post('password')));
        $sql = "SELECT * FROM user WHERE username=? OR email=?";
        $query = $this->db->query($sql, array($data['username'], $data['email']));
        if ($query->num_rows() > 0) return false;
        $this->db->insert("user", $data);
        return true;
    }
}
?>
```
A.3 Views

A.3.1 Head

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>ASOCIACION DE YOGA</title>

<!-- CSS -->
<link rel="stylesheet" href="/css/style.css" type="text/css" media="screen" />
<link rel="stylesheet" href="/css/social-icons.css" type="text/css" media="screen" />
<!-- ENDS CSS -->

<!-- GOOGLE FONTS -->
<link href='http://fonts.googleapis.com/css?family=Ubuntu' rel='stylesheet' type='text/css' />
<!-- ENDS GOOGLE FONTS -->

<!-- JS -->
<script type="text/javascript"
src="/js/jquery-1.5.1.min.js"></script>
<script type="text/javascript"
src="/js/jquery-ui-1.8.13.custom.min.js"></script>
<script type="text/javascript"
src="/js/easing.js"></script>
<script type="text/javascript"
src="/js/jquery.scrollTo-1.4.2-min.js"></script>
<script type="text/javascript"
src="/js/jquery.cycle.all.js"></script>
<script type="text/javascript"
src="/js/custom.js"></script>
<!-- ENDS JS -->

<!-- Isotope -->
<script src="/js/jquery.isotope.min.js"></script>
A.3 Views

A.3.2 Header menu

```html
<body class="home">
  <div id="menu" <!-- Menu -->
    <div class="wrapper" <!-- wrapper-menu -->
      <img src="/img/logo2.png" width=300px align="left" />
      <!-- Navigation -->
      <ul id="nav" class="sf-menu" style="width: 468px; margin-top:12px">
        <li><a href="/">Inicio</a></li>
        <li><a href="#">Bienvenidos</a></li>
        <li><a href="#">Cursos on-line</a></li>
        <li><a href="#">Hatha YOGA</a></li>
        <li><a href="#">Meditación</a></li>
        <li><a href="#">El YOGA terapéutico</a></li>
        <li><a href="#">Reflexología</a></li>
        <li><a href="#">Relajación</a></li>
        <li><a href="#">Control mental</a></li>
        <li><a href="/yoga/index/sala_chat">Filosofía del yoga</a></li>
        <li><a href="/yoga/index/video_gallery">Tantra</a></li>
        <li><a href="#">Contacta con nosotros</a></li>
      </ul>
    </div>
  </div>
</body>
```
<?php
$name = $this -> session -> userdata ( 'name' );
if ( ( ! isset ( $name ) || empty ( $name )) ) {
    echo form_open ( "register/login" );?
    <div class="header-login">
        <div>
            <label for="email" style="color: #FFFFFF"> Email: </label>
            <input type="text" id="email" name="email" value="" size="12" />
        </div>
        <div>
            <label for="password" style="color: #FFFFFF"> Password: </label>
            <input type="password" id="pass" name="pass" value="" size="9" />
        </div>
        <input type="submit" class="" value="Entrar" />
    </div>
    <?php echo form_close (); ?>
</div>
</div>
<div class="header-login id=" boton_registro" style="color:" #FFFFFF">
    <?=php echo anchor ( "register/signin", 'No estas registrado?'); ?>
</div>
<div class="" style="clear"><div></div></div>
</div>
</div>
<!—— ENDS Menu ——>


A.3 Views

A.3.3 Footer

```html
<div id="footer">  <!-- FOOTER -->
  <div class="wrapper">  <!-- wrapper-footer -->
    <ul id="footer-cols">  <!-- footer-columns -->
      <li class="col" style="width:150px;margin-right:40px">
        <h6>Secciones</h6>
        <ul>
          <li class="page_item">
            <a href="/">Inicio</a>
          </li>
          <li class="page_item">
            <a href="#">El centro</a>
          </li>
          <li class="page_item">
            <a href="#">Cursos Online</a>
          </li>
          <li class="page_item">
            <a href="#">Contacta con nosotros</a>
          </li>
        </ul>
      </li>
      <li class="col">
        <h6>Cursos Online</h6>
        <ul>
          <li><a href="#">Hatha YOGA</a></li>
          <li><a href="#">Meditación</a></li>
          <li><a href="#">El YOGA terapéutico</a></li>
          <li><a href="#">Reflexología</a></li>
          <li><a href="#">Relajación</a></li>
          <li><a href="#">Control Mental</a></li>
          <li><a href="#">Tantra</a></li>
        </ul>
      </li>
      <li class="col">
        <h6>La vida es optimismo...</h6>
        "En este mundo traidor nada es verdad ni mentira;
        todo es según el color del cristal con que se mira..."
      </li>
    </ul>
  </div>  <!-- ENDS wrapper-footer -->
</div>  <!-- ENDS FOOTER -->
```
A.3 Views

A.3.4 Home

```html
<div id="slider-block"><!-- Slider -->
  <div id="slider-holder" style="margin-top: 3%; padding-bottom: 2%;">
    <div id="slider">
      <a href="http://www.torrellascarne.com">
        <img src="/images/01.jpg" title="" alt="" /></a>
      <a href="#" img src="/images/02.jpg" /></a>
      <a href="#" img src="/images/03.jpg" /></a>
      <a href="#" img src="/images/04.jpg" /></a>
      <a href="#" img src="/images/06.jpg" /></a>
      <a href="#" img src="/images/07.jpg" /></a>
      <a href="#" img src="/images/08.jpg" /></a>
      <a href="#" img src="/images/09.jpg" /></a>
    </div>
  </div>
</div> <!-- ENDS Slider -->

<!-- MAIN -->

<div id="content" style="padding-top: 0%">
  <ul class="tabs" style="margin: auto;">
    <li><a href="#">Noticias</a></li>
    <li><a href="#">Novedades</a></li>
    <li><a href="#">Comentarios</a></li>
  </ul>
</div> <!-- ENDS MAIN -->
```
A.3 Views

A.3.5 Register view

```php
<div id="content">
  <div class="reg_form">
    <h3 style="padding: 20px 0px 5px 40%;">Registrate</h3>
    <div style="padding-left: 40%; padding-bottom: 20px;">
      Solo son 5 minutos y podras acceder a los contenidos
    </div>
    <?php echo validation_errors('<p class="error">'); ?>
    <?php echo form_open("register/registration"); ?>
    <div style="padding-left: 40%">
      <p><label for="user_name">Nombre de usuario:</label>
        <input type="text" id="user_name" name="user_name" value="<?php echo set_value('user_name'); ?>" /></p>
      <p><label for="email_address">Direccion de correo electronico:</label>
        <input type="text" id="email_address" name="email_address" set_value('email_address'); />
    </div>
    <p><label for="password">Password:</label>
        <input type="password" id="password" name="password" value="<?php echo set_value('password'); ?>" /></p>
    <p><label for="con_password">Confirmacion de Password:</label>
        <input type="password" id="con_password" name="con_password" set_value('con_password'); /></p>
    <p><input type="submit" class="greenButton" value="Proceder" /></p>
  </div>
  <?php echo form_close(); ?>
</div>
```

```php
<form class="reg_form">
  <h3 style="padding: 20px 0px 5px 40%;">Registrate</h3>
  <div style="padding-left: 40%; padding-bottom: 20px;">
    Solo son 5 minutos y podras acceder a los contenidos
  </div>
  <p style="padding-left: 40%; padding-bottom: 20px;">
    <?php echo validation_errors('<p class="error">'); ?>
  </p>
  <?php echo form_open("register/registration"); ?>
  <div style="padding-left: 40%">
    <label for="user_name">Nombre de usuario:</label>
    <input type="text" id="user_name" name="user_name" value="<?php echo set_value('user_name'); ?>" />
  </div>
  <p style="padding-left: 40%; padding-bottom: 20px;">
    <label for="email_address">Direccion de correo electronico:</label>
    <input type="text" id="email_address" name="email_address" set_value('email_address'); />
  </p>
  <p style="padding-left: 40%; padding-bottom: 20px;">
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" value="<?php echo set_value('password'); ?>" />
  </p>
  <p style="padding-left: 40%; padding-bottom: 20px;">
    <label for="con_password">Confirmacion de Password:</label>
    <input type="password" id="con_password" name="con_password" set_value('con_password'); />
  </p>
  <p style="padding-left: 40%; padding-bottom: 20px;">
    <input type="submit" class="greenButton" value="Proceder" />
  </p>
  <?php echo form_close(); ?>
</form>
```
A.3 Views

A.3.6 Thank view

```html
<h1 style="text-align: center; padding: 60px 0px 60px 0px;">Gracias, tu registro ha sido guardado con exito!</h1>

<div style="text-align: center; padding-bottom: 10px">
  <?php echo anchor('yoga','Volver a la pagina de inicio'); ?>
</div>
```

A.3.7 Sala chat view

```html
<!-- content -->
<div id="content">
  <ul>
    <div class="one-half">
      <video class="video" id="centrado" poster="/img/logo2.png" width="500px" height="400px" controls preload>
        <source src="/media/big_buck_bunny2.mp4"></source>
      </video>
      <div id="centrado">
        Descripcion del videos
      </div>
    </div>
    <div class="one-half last">
      <div id="chatbox">
        <?php
        $name = $this->session->userdata('name');
        if (!isset($name) || empty($name))
          echo read_file('./chats/welcome_chat.html');
        else
          echo read_file('./chats/log.html');
        ?>
      </div>
      <input name="usermsg" type="text" id="usermsg" size="63" />
    </div>
  </ul>
</div>
```
A.3 Views

<!— start javascript —>
<script type="text/javascript">
$(document).ready(function(){
    // todo lo que hay aqui abajo se ejecuta una vez el navegador ha recibido todo el HTML (es decir, el documento esta ready)
    $('#usermsg').keypress(function(event){
        if(event.which == 13){
            event.preventDefault();
            var clientmsg = $('#usermsg').val();

            $.post( '<?php echo site_url('/chat/agregar'); ?>', {
                mensaje: clientmsg
            });
            $('#usermsg').attr('value', '');
            loadLog();
        }
    });

    function loadLog(){
        $.ajax({
            type: 'POST',
            url: '<?php echo site_url('/chat/actualizar'); ?>",
            success: function(msg){
                $('#chatbox').html(msg);
                $('#chatbox').animate({ scrollTop: $('#chatbox').height()+999999 }, 'normal');
                // Autoscroll hacia el fondo del div, slow para que vaya lento normal para que vaya mas rapido
            }
        });
    }

    var usermsg = document.getElementById('usermsg');
    <?php $name = $this -> session -> userdata('name');
    if (!isset($name) || empty($name))
        echo 'usermsg.disabled=true; ' ?>
    setInterval(loadLog, 15000);
});
</script>     <!— end javascript —>
<div class="clear"></div>
</div>    <!— ENDS content —>
# Appendix B

## Budget tables

### B.1 First year

**PL**

<table>
<thead>
<tr>
<th>Personnel</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
</table>

**Technical Resources**

<table>
<thead>
<tr>
<th>Technical Resources</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>0</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hosting service</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CDN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SSL Certificate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Office**

<table>
<thead>
<tr>
<th>Office</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bills</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-1.120</td>
</tr>
</tbody>
</table>

**Marketing**

<table>
<thead>
<tr>
<th>Marketing</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Adwords</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-170</td>
</tr>
<tr>
<td>Handouts</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-10</td>
</tr>
</tbody>
</table>

**Earnings**

<table>
<thead>
<tr>
<th>Earnings</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profit rate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**TOTAL P&L**

-43754.46
### B.1 First year

#### CFS

<table>
<thead>
<tr>
<th>Operations</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-3.000</td>
<td>-37.100</td>
</tr>
<tr>
<td>Social security taxes</td>
<td>0</td>
<td>-900</td>
<td>-900</td>
<td>-900</td>
<td>-900</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
<td>-600</td>
</tr>
<tr>
<td>Technical Resources</td>
<td>0</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-248</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-554</td>
</tr>
<tr>
<td>Rental</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-250</td>
<td>-1.120</td>
</tr>
<tr>
<td>Billboard</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-1.120</td>
</tr>
<tr>
<td>Marketing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-248</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>-1.120</td>
</tr>
<tr>
<td>Google Adwords</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td>Handouts</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
</tr>
<tr>
<td>Earnings</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Profit rate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VAT (21%)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Output VAT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Input VAT</td>
<td>-325.5</td>
<td>-77.7</td>
<td>-73.5</td>
<td>-73.5</td>
<td>-159.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-117.1</td>
<td>-1.486</td>
</tr>
<tr>
<td>VAT payable</td>
<td>0</td>
<td>0</td>
<td>476.7</td>
<td>0</td>
<td>306.13</td>
<td>0</td>
<td>351.401</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.134</td>
</tr>
</tbody>
</table>

#### Invest

<table>
<thead>
<tr>
<th>Shareholder investment</th>
<th>50.000</th>
</tr>
</thead>
</table>

#### Balance Sheet

<table>
<thead>
<tr>
<th>Non-current asset</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop</td>
<td>1.200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.200</td>
</tr>
<tr>
<td>Current asset</td>
<td>CFS</td>
<td>5.474</td>
<td>5.474</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Net assets</th>
<th>Shareholder investment</th>
<th>50.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>P&amp;L</td>
<td>-43.754,5</td>
<td>-43.754,5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-current liabilities</th>
<th>Social security taxes</th>
<th>600,0</th>
<th>600,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAT duty</td>
<td>-351,4</td>
<td>-351,4</td>
<td></td>
</tr>
<tr>
<td>ASSET</td>
<td>6.494</td>
<td>6.494</td>
<td></td>
</tr>
<tr>
<td>LIABILITIES</td>
<td>6.494</td>
<td>6.494</td>
<td></td>
</tr>
</tbody>
</table>
**B.2 Second year**

### PL

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social security and other taxes</td>
<td>-600</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
</tr>
</tbody>
</table>

**CFS**

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Social security taxes</td>
<td>-600</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td>-615</td>
<td></td>
</tr>
<tr>
<td>Social security previous year</td>
<td>-600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Technical Resources</strong></td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td>-71</td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Hosting service</td>
<td>-200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDN</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td>-63</td>
<td></td>
</tr>
<tr>
<td>SSL Certificate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><strong>Bills</strong></td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td><strong>Marketing</strong></td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td><strong>Google Adwords</strong></td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td>-150</td>
<td></td>
</tr>
<tr>
<td><strong>Handouts</strong></td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td><strong>Earnings</strong></td>
<td>4.095</td>
<td>5.733</td>
<td>5.323</td>
<td>4.914</td>
<td>4.504</td>
<td>5.733</td>
<td>4.914</td>
<td>4.504</td>
<td>4.914</td>
<td>4.504</td>
<td>4.914</td>
<td></td>
</tr>
<tr>
<td>Profit rate</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL P&amp;L</strong></td>
<td>19.273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Closings

- **B.2 Second year**
- **PL**
- **CFS**
- **Invest**
- **Financial**
  - Previous CFS: 5.474
  - Output VAT: 859.95
  - Input VAT: -121,989
  - VAT payable: 351,401
  - CLOSING CASH: 7.478
  - 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Balance Sheet

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-current asset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laptop</td>
<td>1.020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.020</td>
</tr>
<tr>
<td><strong>Current asset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFS</td>
<td>28.023</td>
<td>28.023</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Net assets</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shareholder investment</td>
<td>50.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50.000</td>
</tr>
<tr>
<td>P&amp;L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.272</td>
</tr>
<tr>
<td>Previous P&amp;L</td>
<td>-43.754</td>
<td>-43.754</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-43.754</td>
</tr>
<tr>
<td><strong>Non-current liabilities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Current liabilities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Social security taxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>615</td>
</tr>
<tr>
<td>VAT duty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.729</td>
</tr>
<tr>
<td><strong>ASSET</strong></td>
<td>28.863</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LIABILITIES</strong></td>
<td>28.863</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### B.3 Third year

#### PL

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Salary</strong></td>
<td>-2,050</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-27,011</td>
</tr>
<tr>
<td><strong>Social security taxes</strong></td>
<td>-615</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-6,150</td>
</tr>
<tr>
<td><strong>Laptops amortization</strong></td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-25,0</td>
<td>-300</td>
</tr>
<tr>
<td><strong>Technical Resources</strong></td>
<td>-74,2</td>
<td>-94,2</td>
<td>-74,2</td>
<td>-274,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
<td>-74,2</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>0</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Hosting service</strong></td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td><strong>SSL Certificate</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Bills</strong></td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-1,000</td>
</tr>
<tr>
<td><strong>Marketing</strong></td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-1,680</td>
</tr>
<tr>
<td><strong>Google Adwords</strong></td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-1,380</td>
</tr>
<tr>
<td><strong>Handouts</strong></td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-100</td>
</tr>
<tr>
<td><strong>Earnings</strong></td>
<td>5,937,8</td>
<td>7,575,8</td>
<td>7,166,3</td>
<td>6,756,8</td>
<td>7,166,3</td>
<td>6,347,3</td>
<td>5,937,8</td>
<td>5,937,8</td>
<td>7,575,8</td>
<td>7,166,3</td>
<td>6,756,8</td>
<td>6,347,3</td>
</tr>
<tr>
<td><strong>Profit rate</strong></td>
<td>1,45</td>
<td>1,85</td>
<td>1,75</td>
<td>1,65</td>
<td>1,75</td>
<td>1,55</td>
<td>1,45</td>
<td>1,45</td>
<td>1,85</td>
<td>1,75</td>
<td>1,65</td>
<td>1,55</td>
</tr>
<tr>
<td><strong>TOTAL P&amp;L</strong></td>
<td>32,089</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### CFS

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Salary</strong></td>
<td>-2,050</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-2,701</td>
<td>-27,011</td>
</tr>
<tr>
<td><strong>Social security taxes</strong></td>
<td>-615</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-810</td>
<td>-6,150</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>0</td>
<td>-20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Hosting service</strong></td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td><strong>SSL Certificate</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Bills</strong></td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-100</td>
<td>-1,000</td>
</tr>
<tr>
<td><strong>Marketing</strong></td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-140</td>
<td>-1,680</td>
</tr>
<tr>
<td><strong>Google Adwords</strong></td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-130</td>
<td>-1,380</td>
</tr>
<tr>
<td><strong>Handouts</strong></td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-10</td>
<td>-100</td>
</tr>
<tr>
<td><strong>Earnings</strong></td>
<td>5,938</td>
<td>7,576</td>
<td>7,166</td>
<td>6,757</td>
<td>7,166</td>
<td>6,347</td>
<td>5,938</td>
<td>5,938</td>
<td>7,576</td>
<td>7,166</td>
<td>6,757</td>
<td>6,347</td>
</tr>
<tr>
<td><strong>Profit rate</strong></td>
<td>1,45</td>
<td>1,85</td>
<td>1,75</td>
<td>1,65</td>
<td>1,75</td>
<td>1,55</td>
<td>1,45</td>
<td>1,45</td>
<td>1,85</td>
<td>1,75</td>
<td>1,65</td>
<td>1,55</td>
</tr>
<tr>
<td><strong>VAT(21%)</strong></td>
<td>1246,928</td>
<td>1590,908</td>
<td>1504,9</td>
<td>1418,9</td>
<td>1504,9</td>
<td>1332,9</td>
<td>1246,9</td>
<td>1246,9</td>
<td>1590,879</td>
<td>1504,91</td>
<td>1418,9175</td>
<td>1332,9235</td>
</tr>
<tr>
<td><strong>VAT payable</strong></td>
<td>-2729,85</td>
<td>0</td>
<td>0</td>
<td>-3983,1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-14,302</td>
</tr>
<tr>
<td><strong>Invest</strong></td>
<td>Laptop</td>
<td>-800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Financial</strong></td>
<td>Previous CFS</td>
<td>28,023</td>
<td>28,023</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CLOSING CASH</strong></td>
<td>28,330</td>
<td>33,474</td>
<td>37,951</td>
<td>37,707</td>
<td>42,184</td>
<td>45,670</td>
<td>44,801</td>
<td>47,791</td>
<td>52,764</td>
<td>53,511</td>
<td>57,493</td>
<td>60,979</td>
</tr>
</tbody>
</table>
### Balance Sheet

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
<th>August</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>Total Year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-current asset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laptops</td>
<td>1,640</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,640</td>
</tr>
<tr>
<td><strong>Current asset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60,979</td>
</tr>
<tr>
<td>CFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60,979</td>
</tr>
<tr>
<td><strong>Net assets</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shareholder investment</td>
<td>50,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50,000</td>
</tr>
<tr>
<td>P&amp;L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32,089</td>
</tr>
<tr>
<td>Previous P&amp;L</td>
<td>-24,482</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-24,482</td>
</tr>
<tr>
<td><strong>Non-current liabilities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Current liabilities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Social security taxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>810</td>
</tr>
<tr>
<td>VAT duty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3,901</td>
</tr>
<tr>
<td><strong>ASSET</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62,319</td>
</tr>
<tr>
<td><strong>LIABILITIES</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62,319</td>
</tr>
</tbody>
</table>
Bibliography


http://www.coralcdn.org/.


http://www.captcha.net/.