UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ

SPECIALIZAREA INFORMATICǍ

LUCRARE DE LICENŢĂ

Studentul Cǎlǎtor

Conducător ştiinţific

Lector Universitar, Dr. DARABANT Sergiu Adrian

Absolvent

Cantor Vlad Mihai

**2015**

BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

TRAVELER STUDENT

Supervisor

Lecturer Professor Dr. DARABANT Sergiu Adrian

Author

**2015**

### Table of Contents

[Table of Contents 5](#_Toc421496617)

[Chapter 1. Introduction. 7](#_Toc421496618)

[Chapter 2. .NET Framework and C# Language. 10](#_Toc421496619)

[2.1 .NET Framework 10](#_Toc421496620)

[2.1.1 Common Language runtime 10](#_Toc421496621)

[2.1.2 Class Library 11](#_Toc421496622)

[2.2 C# Language 11](#_Toc421496623)

[2.3 Universal Windows Applications. 12](#_Toc421496624)

[2.4 Web services. 12](#_Toc421496625)

[2.5 Microsoft SQL Server. 12](#_Toc421496626)

[Chapter 3. Streaming media. 12](#_Toc421496627)

[3.1 Internet Information Services 15](#_Toc421496628)

[3.2.1 IIS Media Services and Smooth Streaming 15](#_Toc421496629)

[Smooth Streaming Disk File Format 16](#_Toc421496630)

[Smooth Streaming Wire File Format 17](#_Toc421496631)

[Chapter 4. Traveler Student application. 18](#_Toc421496632)

[4.1 Use cases 18](#_Toc421496633)

[4.1.1 Teacher related use cases 19](#_Toc421496634)

[1) View Taught Courses 19](#_Toc421496635)

[2) Pick Course 19](#_Toc421496636)

[3) Begin Course 19](#_Toc421496637)

[4) Pause course 20](#_Toc421496638)

[5) End Course 20](#_Toc421496639)

[6) Switch mode 20](#_Toc421496640)

[7) View Chat Room 20](#_Toc421496641)

[8) Respond Messages 20](#_Toc421496642)

[9) View Course Quizzes 20](#_Toc421496643)

[10) Send Quiz 20](#_Toc421496644)

[11) View All Questions 20](#_Toc421496645)

[12) Add Questions 20](#_Toc421496646)

[13) Pick Questions for Quiz 21](#_Toc421496647)

[14) Save Quiz 21](#_Toc421496648)

[7.1.2 Student related use cases 21](#_Toc421496649)

[1) View Attended Courses 21](#_Toc421496650)

[2) Pick Course 21](#_Toc421496651)

[3) View Chat Room 21](#_Toc421496652)

[4) Ask Question 21](#_Toc421496653)

[5) View Available Quizzes 22](#_Toc421496654)

[6) Answer Quiz 22](#_Toc421496655)

[7) Enter Course 22](#_Toc421496656)

[8) Leave course 22](#_Toc421496657)

[7.1.3 Administrator related use cases 22](#_Toc421496658)

[1) Add Teacher 23](#_Toc421496659)

[2) Add Student 23](#_Toc421496660)

[3) View Courses 23](#_Toc421496661)

[4) Add Course 23](#_Toc421496662)

[5) Pick Course 23](#_Toc421496663)

[7) Set teacher 23](#_Toc421496664)

[8) View All Teachers 24](#_Toc421496665)

[9) Subscribe Student 24](#_Toc421496666)

[10) View All Students 24](#_Toc421496667)

[12) Remove student 24](#_Toc421496668)

[4.2 Architecture 24](#_Toc421496669)

[4.2.1 Model Layer 25](#_Toc421496670)

[4.2.2 Repository Layer 25](#_Toc421496671)

[4.2.3 Business Logic Layer 26](#_Toc421496672)

[4.2.4 Services Layer 27](#_Toc421496673)

[4.2.5 Client Controller 27](#_Toc421496674)

[4.2.6 User Interface 27](#_Toc421496675)

[4.3 Database 27](#_Toc421496676)

[Chapter 5.Conclusions. 28](#_Toc421496677)

[Chapter 6.Bibliografy 28](#_Toc421496678)

## Chapter 1. Introduction.

The main idea of this diploma thesis is to bring together the e-learning concept and livestream in order to simulate the exact conditions of a classroom. This project wants to help students that follow a distance studies program, by allowing them to interact with their teacher. Also it will offer teachers who are travelling a lot the possibility to keep their curse even if they are on the other half of the world. An important thing is that usability of this platform may extend without modifications of the source code because even if the main purpose is to serve students and teachers, it can be used for any webinar or training session.

Beside the main feature, which is livestreaming, there are a set of extra features that make this platform more attractive. An important feature is allowing a questions and answers session, where students can put some questions which can be either private, only with the teacher, or in the chat room made with all students that have subscribed for the current curse. To this questions it will be teacher’s choice when to answer to that question, he can respond instantly, or when he thinks is properly to do it.

A feature that is useful for students is that curses are also persisted and they can be replayed any time the user wants. This feature may come in handy when the user wants to recapitulate things that have been discussed at the curse or they cannot be online when the curse is kept. In this case the chat feature will not be available.

In order to increase interactivity of the curse beside the chat, the teacher can initiate a quick quiz, which can have a time limit per question or per entire quiz. The students will not be able to make anything else with the application during a test. The test will be structured as a multiple choice and will be automatically corrected when the user will finish it or when the time expires. Only the teacher will be able to see the results of the test.

In order to keep track of the users that are present at a certain curse, the application will be able mark as present both users that are connected remote to the curse and those who have a smart phone with the mobile application for remote learning installed on it. This information can be used to send quizzes or in statistical purposes.

Another thing worth mentioning is that the platform will contain a Windows Phone 8.1 and a Windows 8.1 application, which will offer students a user-friendly experience, an application on the same platforms for the teacher and a server which will serve applications with requested data.

This chapter will also make a resume for every theme that will be covered by this thesis.

The entire implementation is based on .NET framework, so it is compulsory to talk in the first part after the introduction about this framework considering that all chapters that will follow this one will be strongly related to it. I will try to make a brief presentation of the main concepts that this framework facilitates.

The next chapter, named Streaming, will present the technologies and frameworks used in order to create the livestream environment and also how video data will be persisted. It will explain the smooth streaming concept, will offer an overview of the software development kit that enforces this concept and will discuss about encoding and types of encoding required by smooth streaming. Also this division will refer to Internet Information Services and Media Services extra feature needed in order to make livestream and on demand stream available.

In the chapter Universal Windows Applications we will discuss a little bit about the new concept that Microsoft has brought into developers world. I am referring to the fact that you can create an application that can run on phones, tables, laptops and personal computers. Also it will contain some explanations regarding Microsoft Media Platform's Player Framework and Smooth Streaming Client Software Development Kit. Near field communication will be another theme discussed because it is used to signal the presence of students at a certain curse.

The next chapter is about persistence, relational databases and Microsoft SQL Server 2012, which will be used as database management system. It will explain some terms as store procedure and discuss how SQL injection is avoided using their parameterized version.

Another chapter will refer to web services in order to make persisted data accessible. Here we will talk about Simple Object Access Protocol (SOAP), its benefits and downsides, also about how they are implemented in .NET framework.

If until now we have presented all the technologies used in order to create the application, in the next part a detailed overview of it will be made. Here we will talk about architecture, detailed feature explanation and implementation details. In this chapter will exist the use case diagram of the application and each use case will be explained. For each use case will exist a sequence diagram which will show the flow of data and the execution flow. Also will exist a class diagram which will help as observe what design patterns have been used and make an idea how applications are implemented. A database diagram will be added in order to easily understand how data is persisted. Main algorithms will also be presented in this area and will be explained.

The last chapter will contain the conclusions over this thesis.

There are a lot of e-learning platforms, which offer functionalities similar with Traveler Student. As a first example let’s look into Harvard Extension School, which is a project that offers videotaped, live web-conferences and hybrid courses. A big advantage is that people from all around the world can join this classes without travelling to the campus. A difference between their project and Traveler Student is the quiz feature, which provides a way to examine students and also to verify the attention to the current course.

Another online courses provider is Pluralsight. This is a web only platform that offers videotaped lectures for developers and IT admins. They have a very large number of courses and they cover almost all technologies and frameworks. The interesting part is that videos are based on examples and cover details with respect to the level that has been assigned to that course. A nice feature is that they mark the videos that you have seen, an idea that Traveler Student will inherit and improve it by saving the courses you have attended from the classroom.

## Chapter 2. .NET Framework and C# Language.

### 2.1 .NET Framework

.Net Framework is a technology that supports building and running applications of many types and XML Web Services. It can be split in two big parts: first one is the common language runtime and the second one is class library.

#### 2.1.1 Common Language runtime

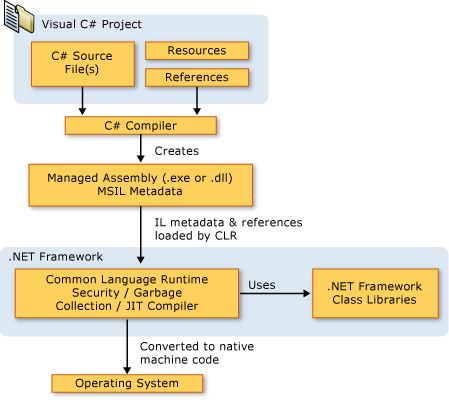
We can say that Common language runtime is the core component of the framework. It manages memory, thread execution, code execution, code safety verification, compilation and other services. It supports many languages, for example: C++, Visual Basic and F#, but the discussion will be more about C#, given that all applications developed for this thesis are written in this language. In other words this component takes code which comes as Microsoft Intermediate Language (MSIL) metadata and transforms it into machine code that will be executed by the processor. The figure below shows the lifecycle of the source code.

Figure 1 [https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx]

An important feature of the above mentioned component is memory management, also known as Garbage Collector. It is used in order to identify and free unused allocated memory. Garbage collection is a very expensive operation which, if is made at the wrong moment, can decrease drastically the performance of a program. So in order to avoid that there are three situations in which it is realized: when the system is low on physical memory, when the allocated memory surpasses a certain limit, which is adjusted as the process runs and when the GC.Collect method is called, situation which is totally unadvised.

#### 2.1.2 Class Library

.NET Framework includes, as you can see in the schema above, has a component, named Class Library, which includes a set of standard libraries. They are organized, hierarchical, in namespaces. For most of the Application Programming Interfaces included in it the root namespaces are System and Microsoft. This library contains a large number of useful functions such as graphic rendering, file reading and writing, XML manipulation, and many others. Also it can be divided into two parts: Framework Class Library and Base Class Library.

Base Class Library is the main core part of the framework. It includes base types for objects and important components which allow Common Language Runtime to operate properly. It includes namespaces as: System, System. Collections, System. Collections.Generic, System.Diagnostics, System.Globalization, System.IO, System.Security, System.Security.Permissions and System.Text, System.Threading.

Framework Class Library includes an expanded set of libraries as Windows Forms, ADO.NET, ASP.NET, Language Integrated Query (LINQ), Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF) and Workflow Foundation (WF). It is very vast, being comparable as dimension with Java standard libraries.

### 2.2 C# Language

C# is a C derived object-oriented language, which enforces strong typing. It supports all object oriented concepts as encapsulation, inheritance and polymorphism, but has some extra features that make it easier to develop software. It is Microsoft programming language developed to compete with Sun’s Java language.

Even if it is a language that is like C++ and Java there are some interesting, innovative and handful language constructs. Delegates is one of them, which are encapsulated method signatures, allowing type-safe event notifications. Another one is the existence of so named Properties which are members with getter and setter either default defined or customized. Attributes are much alike Java annotations and are used to provide declarative metadata about types at runtime. A very interesting feature is Language-Integrated Query (LINQ) which allows you to make SQL alike queries, but over System.Collections data structures.

If until now the discussion was about differences it is important to underline that Java and C# are very much alike. An important similarity is that both are using interfaces for multiple inheritance, which is different from C++ way things work. Another similitude is the fact that any variable and method have to be encapsulated in a class, in other words there are no global variables and methods.

### 2.3 Universal Windows Applications.

“A Universal Windows app is a Windows experience that is built upon the Universal Windows Platform (UWP), which was first introduced in Windows 8 as the Windows Runtime. Universal Windows apps are most often distributed via the Windows Store (but can also be side-loaded), and are most often packaged and distributed using the .APPX packaging format.”[2]

### 2.4 Web services.

“The infrastructure for Web services is built to conform to industry standards such as SOAP, XML, and WSDL, and this allows clients from other platforms to interoperate with Web services. As long as a client can send standards-compliant SOAP messages, formatted according to a service description, that client can call a Web service created using ASP.NET (regardless of the platform on which the client resides).”[2]

### 2.5 Microsoft SQL Server.

## Chapter 3. Streaming media.

Streaming media refers to the process of constantly receiving multimedia from a provider and presenting it to the end-user. The difference between streaming multimedia and downloading it is that when you stream it you can play it before the entire file has been stored on your local drive.

The first patented system for transmission and distribution of signal over electrical lines was designed by Major General George Owen Squier the early 1920s. His patents were acquired by North American Company in 1922, which created Wired Radio Inc., a company that delivered music, charging for it on the electric bill.

Even if the technology was discovered early, streaming media over computer networks was not possible until June 24, 1993, when the band “Severe Tire Damage” performed live on the Internet for the first time.

Live streaming is a division of multimedia streaming, which refers to delivering live, over the internet, the data received from a source of media (for example a video camera or a voice recorder). The process is a bit more complex than it is presented in the above definition, considering that the content is encoded and several protocols are applied over it before it is distributed to the client, which will unpack, decode and then present it.

There are many general delivery methods. Most important are: traditional streaming, progressive download, and adaptive streaming.

A big problem that concerns streaming media is that it has to handle big amount of data. An improvement that would reduce the dimension of data would be to compress it. Here intervene the so named codecs. A codec, in our case, is a computer program, which is concerned with encoding-decoding and compressing-decompressing data. There are two types of codecs: video and audio. For each one there are multiple compression formats.

Real-Time Streaming Protocol (RTSP) is a good example of a traditional streaming protocol. RTSP is defined as a stateful protocol, which means that from the first time a client connects to the streaming server until the time it disconnects from the streaming server, the server keeps track of the client's state. The client communicates its state to the server by issuing it commands such as PLAY, PAUSE or TEARDOWN (the first two are obvious; the last one is used to disconnect from the server and close the streaming session).

After a session between the client and the server has been established, the server begins sending the media as a steady stream of small packets (the format of these packets is known as RTP). The size of a typical RTP packet is 1452 bytes, which means that in a video stream encoded at 1 megabits per second (Mbps), each packet carries approximately 11 milliseconds of video. In RTSP the packets can be transmitted over either UDP or TCP transports—the latter is preferred when firewalls or proxies block UDP packets, but can also lead to increased latency (TCP packets are re-sent until received).

Adaptive streaming is a hybrid delivery method that acts like streaming but is based on HTTP progressive download. It's an advanced concept that uses HTTP rather than a new protocol.

Another common form of media delivery on the Web today is progressive download, which is nothing more than a simple file download from an HTTP Web server. Progressive download is supported by most media players and platforms, including Adobe Flash, Silverlight, and Windows Media Player. The term "progressive" stems from the fact that most player clients allow the media file to be played back while the download is still in progress—before the entire file has been fully written to disk (typically to the Web browser cache). Clients that support the HTTP 1.1 specification can also seek to positions in the media file that haven't been downloaded yet by performing byte range requests to the Web server (assuming that it also supports HTTP 1.1).

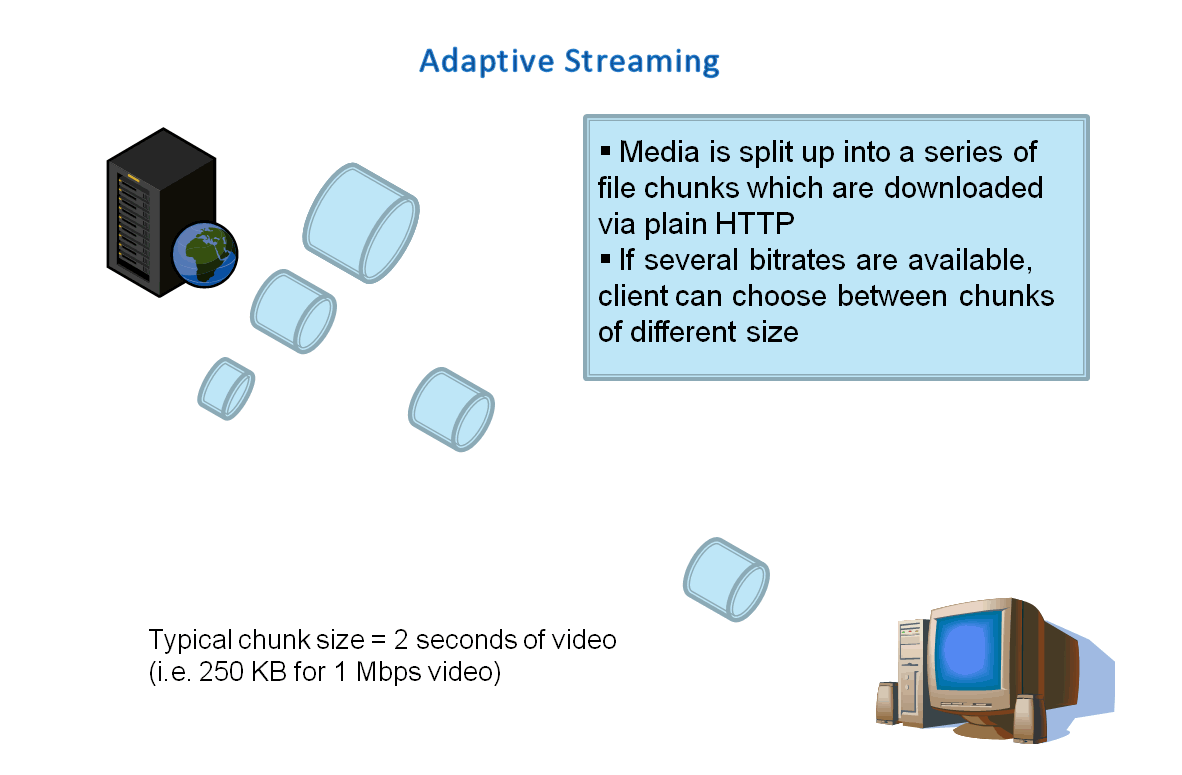
Popular video sharing Web sites on the Web today, including YouTube, Vimeo, MySpace, and MSN Soapbox, almost exclusively use progressive download.

Unlike streaming servers that rarely send more than 10 seconds of media data to the client at a time, HTTP Web servers keep the data flowing until the download is complete. If you pause a progressively downloaded video at the beginning of playback and then wait, the entire video will eventually have downloaded to your browser cache, allowing you to smoothly play the whole video without any hiccups.

There is a downside to this behavior as well—if 30 seconds into a fully downloaded 10 minute video, you decide that you don't like it and quit the video, both you and your content provider have just wasted 9 minutes and 30 seconds worth of bandwidth.

In a typical adaptive streaming implementation, the video/audio source is cut into many short segments ("chunks") and encoded to the desired delivery format. Chunks are typically 2-to-4-seconds long. At the video codec level, this typically means that each chunk is cut along video Group of Pictures (GOP) boundaries (each chunk starts with a key frame) and has no dependencies on past or future chunks/GOPs. This allows each chunk to later be decoded independently of other chunks.

The encoded chunks are hosted on a HTTP Web server. A client requests the chunks from the Web server in a linear fashion and downloads them using plain HTTP progressive download. As the chunks are downloaded to the client, the client plays back the sequence of chunks in linear order. Because the chunks are carefully encoded without any gaps or overlaps between them, the chunks play back as a seamless video.

The "adaptive" part of the solution comes into play when the video/audio source is encoded at multiple bit rates, generating multiple chunks of various sizes for each 2-to-4-seconds of video. The client can now choose between chunks of different sizes. Because Web servers usually deliver data as fast as network bandwidth allows them to, the client can easily estimate user bandwidth and decide to download larger or smaller chunks ahead of time. The size of the playback/download buffer is fully customizable.

### 3.1 Internet Information Services

“Internet Information Services (IIS) for Windows® Server is a flexible, secure and manageable Web server for hosting anything on the Web. From media streaming to web applications, IIS’s scalable and open architecture is ready to handle the most demanding tasks.” [3].

### 3.2.1 IIS Media Services and Smooth Streaming

IIS Media Services integrates with Internet Information Services (IIS) to provide a secure, scalable, and easy-to-manage HTTP-based media delivery platform for managing rich and dynamic websites, applications, and services.

This extension has many features but we are interested in two of them: Smooth Streaming and Live Smooth Streaming.

Smooth Streaming enables HTTP adaptive streaming of media to Silverlight clients over HTTP. Smooth Streaming provides a high-quality viewing experience that adapts to actual bandwidth and video rendering conditions on the client. You can use this to deliver on-demand video in true HD 1080p to customers with enough bandwidth and modern hardware.

Live Smooth Streaming enhances the Smooth Streaming extension by enabling HTTP adaptive streaming of live media events to Silverlight clients over HTTP. Live Smooth Streaming enables client-side features such as Instant Replay, Slow Motion, Multiple Camera Angles, and Live Ad Integration, and provides a high-quality viewing experience that scales massively over existing content distribution networks to make delivery of live HD video to large audiences a reality.

IIS Smooth Streaming uses the MPEG-4 Part 14 (ISO/IEC 14496-12) file format as its disk (storage) and wire (transport) format. Specifically, the Smooth Streaming specification defines each chunk/GOP as an MPEG-4 Movie Fragment and stores it within a contiguous MP4 file for easy random access. One MP4 file is expected for each bit rate. When a client requests a specific source time segment from the IIS Web server, the server dynamically finds the appropriate Movie Fragment box within the contiguous MP4 file and sends it over the wire as a standalone file, thus ensuring full cacheability downstream.

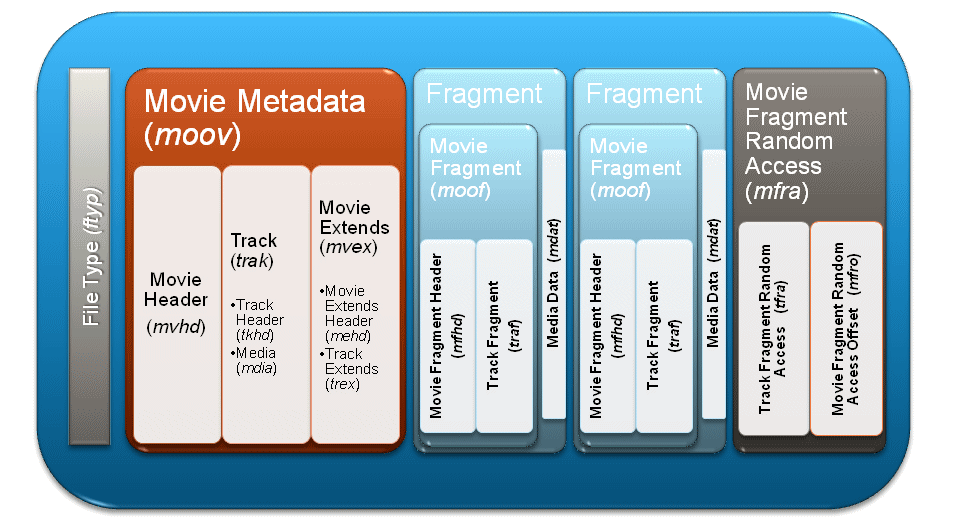
In other words, with Smooth Streaming, file chunks are created virtually upon client request, but the actual video is stored on disk as a single full-length file per encoded bit rate. This offers tremendous file-management benefits.

Smooth Streaming is based on the ISO/IEC 14496-12 ISO Base Media File Format specification, better known as the MP4 file specification.

There are actually two parts to the Smooth Streaming format: the wire format, and the disk file format. In Smooth Streaming, a video is recorded in full length to the disk as a single file (one file per encoded bit rate), but it's transferred to the client as a series of small file chunks. The wire format defines the structure of the chunks that are sent by IIS to the client, whereas the file format defines the structure of the contiguous file on disk, enabling better file management. Fortunately, the MP4 specification allows MP4 to be internally organized as a series of fragments, which means that in Smooth Streaming the wire format is a direct subset of the file format.

#### Smooth Streaming Disk File Format

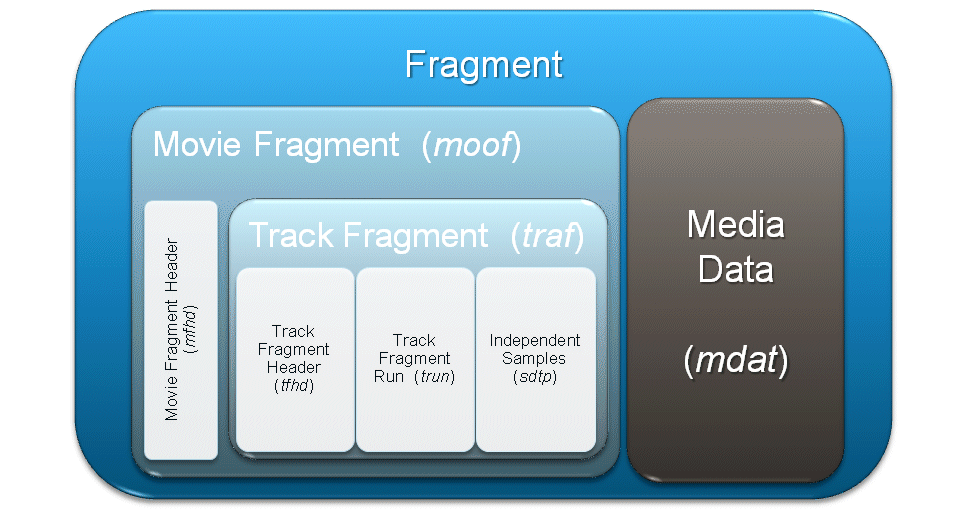
The basic unit of an MP4 file is called a "box." These boxes can contain both data and metadata. The MP4 specification allows for various ways to organize data and metadata boxes within a file. In most media scenarios, it's considered useful to have the metadata written before the data so that a player client application can have more information about the video/audio that it's about to play before it plays it. However, in live streaming scenarios it's often not possible to write the metadata up-front about the whole data stream because it's simply not yet fully known. Furthermore, less up-front metadata means less overhead, which can lead to shorter startup times. For these reasons the MP4 ISO Base Media File Format specification is designed to allow MP4 boxes to be organized in a fragmented manner, where the file can be written "as you go" as a series of short metadata/data box pairs, rather than one long metadata/data pair.

The following figure is a high-level overview of what a Smooth Streaming file looks like on the inside:

In a nutshell, the file starts with file-level metadata ('moov') that generically describes the file, but the bulk of the payload is actually contained in the fragment boxes that also carry more accurate fragment-level metadata ('moof') and media data ('mdat'). (The above diagram only shows 2 fragments, but a typical Smooth Streaming file has a fragment for each 2 seconds of video/audio.) Closing the file is an 'mfra' index box that allows easy and accurate seeking within the file.

#### Smooth Streaming Wire File Format

When a player client requests a video time slice from the IIS Web server, the server seeks to the appropriate starting fragment in the MP4 file and then lifts the fragment out of the file and sends it over the wire to the client. This is why we refer to the fragments as the "wire format." This technique greatly enhances the efficiency of the IIS Web server because it doesn't induce any remuxing or rewriting overhead.

The following figure shows what an MP4 fragment looks like in more detail:

Within the guidelines of the MP4 ISO Base Media File Format specification, the Smooth Streaming format uses a custom box organization schema and some custom boxes. To differentiate Smooth Streaming files new file extensions have been created: \*.ismv (video and audio) and \*.isma (audio only).

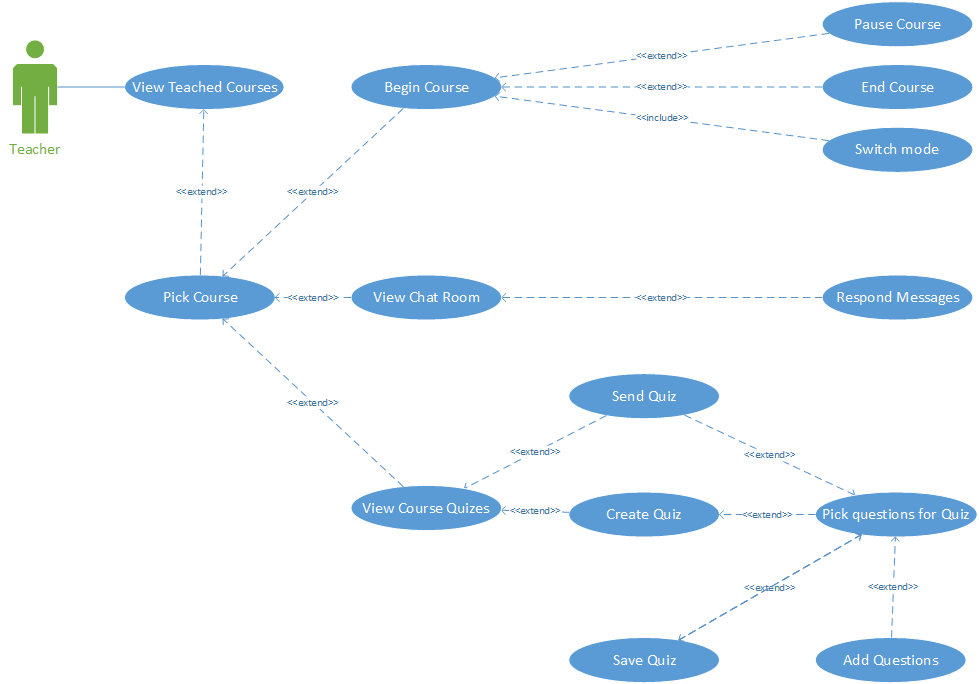
## Chapter 4. Traveler Student application.

As mentioned in the Introduction chapter, this chapter will discuss about TravelerStudent’s architecture details and use cases. A detailed presentation of how this platform can be used will be made first and then the parts about its core and code behind.

### 4.1 Use cases

This platform has three possible actors: Teacher, Student and Administrator. Teacher is the user that will present the course, that would interact with use cases like Quiz creation and course presentation. In order to make Use case diagrams more readable the LogIn use case has been excluded because this platform does not allow any anonymus operations. It should be included as the first use case for every actor.

#### 4.1.1 Teacher related use cases

The teacher is the user with the most use cases, so the diagram below offers an overview of all of them and the relations that they enforce.

##### View Taught Courses

After the user have logged in he will see all the courses that he is responsible for. This use case is necessary because a lecturer may preach more than one subject.

##### Pick Course

This actor should be able to choose the lecture that he wants to work with. There no other actions possible without selecting an item from the list. This action is enforced because there are no use cases that involve more than one course.

##### Begin Course

Represents the beginning of the broadcast, in other words the moment when the teacher starts the audio and video streaming from the course. In this moment all the students are enrolled for this subject will receive a notification that the course have begun.

##### Pause course

The lecturer should have the possibility to pause the transmission without ending the course.

##### End Course

This one gives the actor the possibility to stop the streaming and mark the end of the lesson. The broadcast will be stopped and student will be notified that current lecture has ended.

##### Switch mode

The teacher is able to stream images from the class room and from the desktop of the computer he uses. This use case allows him to change the source of the images.

##### View Chat Room

TravelerStudent offers a chat room for every course so the teacher is able to read those messages.

##### Respond Messages

The chat room is available when the transmission is off so the teacher must have a way to respond questions that are posted there. Here intervenes this use case which allow him to provide the information that the students requested. In order to keep an academic air no private messages are allowed.

##### View Course Quizzes

A professor is able to give students tests with this platform. This quizzes have been predefined by him and this use case offers the opportunity to view all of them.

##### Send Quiz

The actor can send the test to all users that have attended this course. This use case can be reached in two ways. One is by picking a test from the list and the other is by creating a new one and then sending it.

##### View All Questions

This functionality offers the possibility of viewing all questions available for the current course.

##### Add Questions

Adding a question assumes to enter the problem statement and all the choices that will be available. Considering that only multiple choice questions are allowed, the user also need to provide the correct answer. It is needed for the automatic correction of the tests. Another information requested is the average time that the teacher considers proper in order to answer to the requirement.

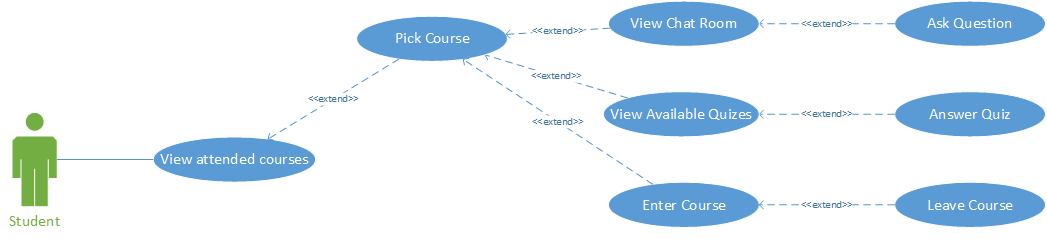
##### Pick Questions for Quiz

From the list with all the questions the lecturer has previously added he can add the ones that he considers fitted in. He can also pick the way time is counted: per question or per quiz as a sum of time limits from all questions.

##### Save Quiz

If the teacher intends to reuse the quiz he has just created, then he can save it.

#### 7.1.2 Student related use cases

 Another actor, the user which is going to use the application in order to watch and learn, does not have so many use cases. In a few words he is able only to watch, chat and answer quizzes.

##### View Attended Courses

A student should be able to see the list of courses in which he is enrolled, because it is easier to identify the class he wants to attend if he sees only those he joined.

##### Pick Course

Given the fact that a scholar can attend more than one courses all functionalities regarding one lecture will became available only when he has selected it from the list.

##### View Chat Room

This use case represents opening the chat room for a certain lecture.

##### Ask Question

This actor is also able to write in the chat room.

##### View Available Quizzes

If there are quizzes that the current logged in actor have not solved he will be able to see them here. The other functionality offered is that the user can see the results for previous tests at currently selected course here.

##### Answer Quiz

After the student selected the quiz he wants to take the test will begin. If he will leave the test or the time will expire the test will be corrected at current state.

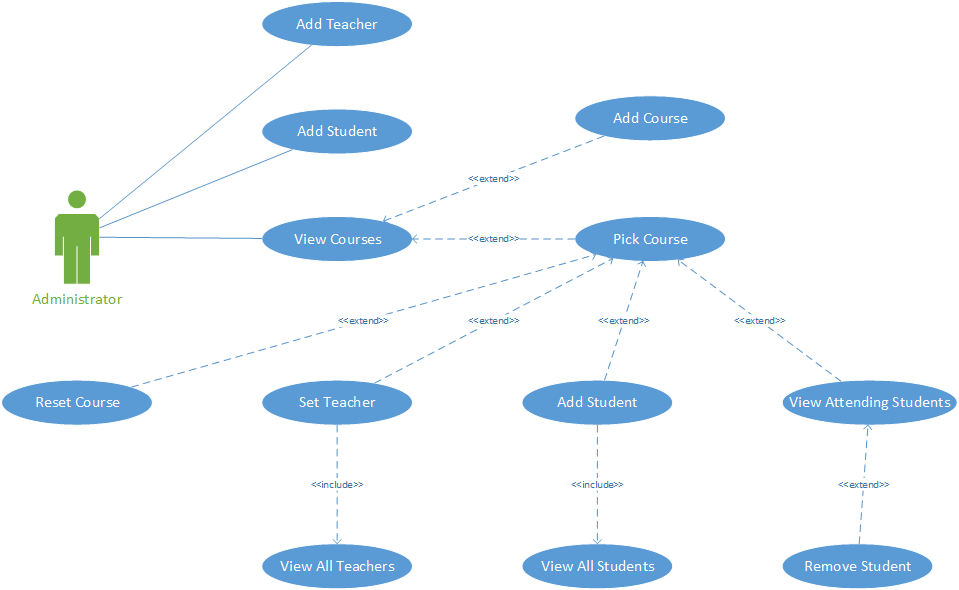
##### Enter Course

The learner will begin watching the live images from the classroom.

##### Leave course

The student will stop watching live the course.

#### 7.1.3 Administrator related use cases

The third actor is the administrator of the system. He is the user that controls the way courses teachers and students are related.

##### Add Teacher

Adding a new teacher is a feature that becomes useful when new lecturers are employed or decide to join the cause.

##### Add Student

Considering that there are no scenarios which do not require log in any student must be have an account.

##### View Courses

The administrator is the only actor that is allowed to see all the existing courses.

##### Add Course

Allows the administrator to add a new subject to the collection.

##### Pick Course

The administrator is the user that is able to make changes in the structure of a course. He can do that for a course at a time so this use case which represents the choosing action.

1. Reset Course

This functionality allows the actor to delete all information related to the chosen course. It will unsubscribe students from the current lecture, will remove quiz results, quizzes, questions, recorded lessons, messages from chatroom.

##### Set teacher

It represents the action of assigning the teacher that will lecture to the current course.

##### View All Teachers

In order to choose the professor that will edify the students the administrator will see the complete list of teachers.

##### Subscribe Student

A student will receive notifications and will be able to attend a lecture only if the administrator has added him to the list of learners.

##### View All Students

An administrator is able to see the complete list of students for choosing which one will attend the current course.

1. View Attending Students

Because he is able to unsubscribe a student at a time the administrator can see the list of users that are following the selected course.

##### Remove student

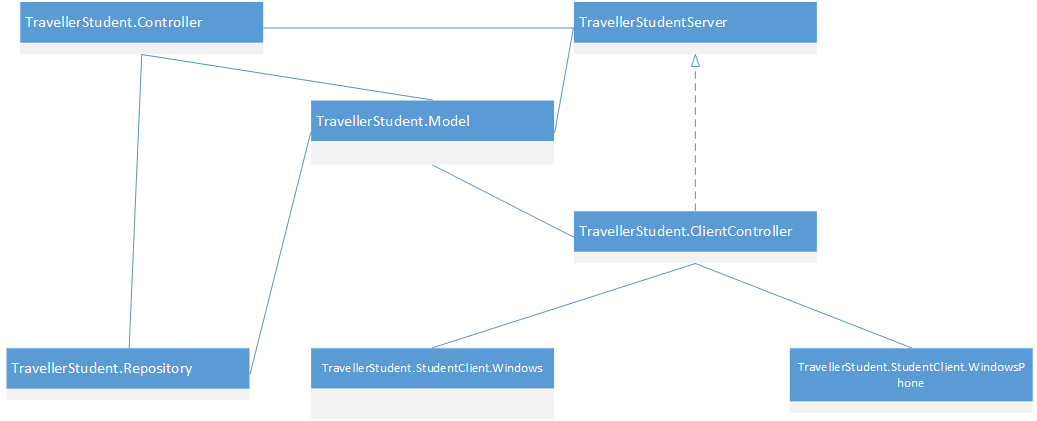
This feature allows the actor to remove a single user from list of attendees.

### 4.2 Architecture

The platform offers a client/server architecture. The client/server architectural style describes distributed systems that involve a separate client and server system, and a connecting network. It can be defined as the relationship between a client and one or more servers, where the client initiates one or more requests (perhaps using a graphical UI), waits for replies, and processes the replies on receipt. The server typically authorizes the user and then carries out the processing required to generate the result. The server may send responses using a range of protocols and data formats to communicate information to the client.

Another design pattern that has been implemented by this platform is model-view-controller. “The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes” [2, https://msdn.microsoft.com]:

* Model: “The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller)” [1, Burbeck].
* View: The view manages the display of information.
* Controller: “The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate” [2, <https://msdn.microsoft.com>].

The solution is organized in multiple projects, each of them representing an important component of the system. The diagram below shows the relationships between them.

The figure above also draws the schema of a layered architecture where each component serves a single role. “Layered architecture focuses on the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Communication between layers is explicit and loosely coupled. Layering your application appropriately helps to support a strong separation of concerns that, in turn, supports flexibility and maintainability.” [2, <https://msdn.microsoft.com>]

Further will be discussed each layer and will be explained its role in the system.

#### 4.2.1 Model Layer

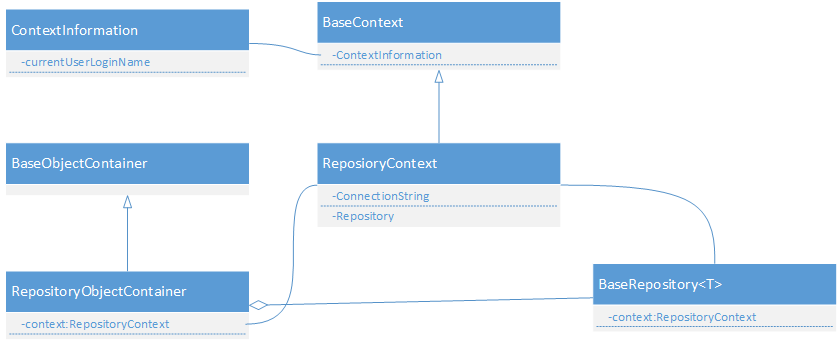
It is named “TravellerStudent.Model” and its concern is to represent data. It is referenced by all the other layers because they work with the notions defined in this one. It also contains the base objects on top of which the other layers are built.

#### 4.2.2 Repository Layer

This level is concerned with interactions with the database. The only thing that it does is to read, insert and update the objects that are mapped in the Model and persist them. This project has also his own architecture. Given the fact that for each class from the model you will find one in this layer that will handle CRUD operations in the database.

In the diagram below you can see that the structure that allows adding as many repositories and using them in the other layers that reference this one only by inheriting BaseRepository and creating an instance of the object in the RepositoryObjectContainer. Because if every repository would be instantiated any time a RepositoryContext would be created a lot of memory would be wasted because is very probably to use only one or two of them at a time. In order to prevent that lazy initialization has been used. Lazy initialization of an object means that its creation is deferred until it is first used. Lazy initialization is primarily used to improve performance, avoid wasteful computation, and reduce program memory requirements.

The BaseRepository contains methods that act as wrappers over the calls to the database and an abstract method that must be implemented by all children in order to map the representations from the database with the ones from model. This method is used when a read call is made.



#### 4.2.3 Business Logic Layer

Its role is to process data received from the user and from the repository layer. From a structural point of view this plan has the same organization as the repository level. A difference is that beside the access to its inner classes through ControllerContext it also has access, through a Repository Context, to the data warehouse. Another change is that in the controller the classes are not oriented only after the model but after functionality too. In other words besides working with the repository controller manages operations over the Media Services from the Internet Information Services.

#### 4.2.4 Services Layer

This is the entry point for the server. Any client will be able to make requests in order to retrieve information, or to send data to be persisted.

Its only purpose is to offer clients access to the business layer. The technology used to support requests from the clients is Simple Object Access Protocol (SOAP). The services in this project are also organized hierarchically because exists a BaseService which checks the validity of all requests, and creates the ControllerContext which will be used by all child classes to call methods from the business layer.

#### 4.2.5 Client Controller

This is a wrapper over the Services Layer, which will have the role of communicating with the server and process data received from it. Its purpose is to create requests and interpret the responses. Also in this layer will be made the last process of data. Here should be made the computations that can be made on the client because they doesn’t need worth requesting the server.

#### 4.2.6 User Interface

This time there are several projects that have that role, considering that there are two applications: one for Windows Phone 8.1 and one for Windows 8.1. Their only concern is to render the user interface and to keep it in sync with the data in the backend.

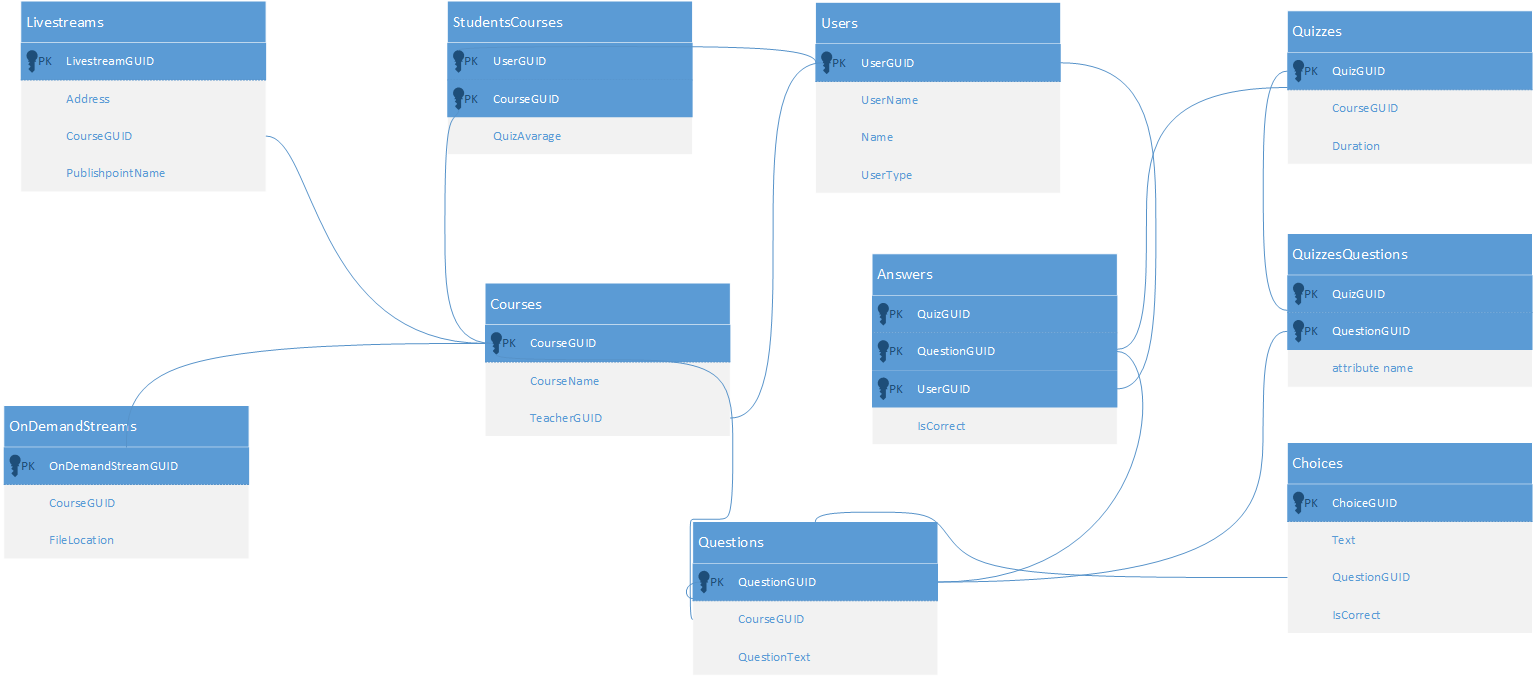
### 4.3 Database

The database persists information regarding users, information about quizzes, the addresses for livestreams and the location of the files where are store courses that expired.

All operations requested by the Repository layer are organized as stored procedures. In other words for each method that makes a call to the data warehouse a stored procedure has been created. This measure has been taken for two reasons: first one is that this is a good way to avoid SQL injection and the second one is to organize SQL commands in order to be easy to modify later.

SQL injection is avoid by parameterizing the stored procedures that query data based on user inputs. Also there are no dynamically created commands in order to make it possible.

This method is also scalable beside flexible because you can add store procedures without altering the ones created until then.

 In the figure below it is presented the stable structure of the database.

### 4.4 Publishing Points Management

Besides processing data from the database, the server, has also the role to administrate the Publishing points and the files for on demand streaming. This operations are realized through the IIS Smooth Streaming Management REST Services Application Programming Interface.

“The Smooth Streaming Management REST Services API provides programmatic access to much of the functionality available through the Live Smooth Streaming Publishing Points icon in the Media Services area of IIS Manager. Using the Smooth Streaming Management REST Services API, you can create or delete publishing points; enumerate the publishing points on a Web site; query the setting, state, or statistics of a publishing point; or update the state of a publishing point.”[2]

This part of the platform is also dependent of the database because the name of the sites. Location of the files and the most important the mapping between them and the lectures are persisted in the database.

### 4.5 Encoding Data

Another missing part from the puzzle is how data is sent to the server and from there to the client. This problem has been solved by using Expression Encoder SDK.

This platform has been developed by Microsoft in order to

## Chapter 5.Conclusions.

## Chapter 6.Bibliografy

1. Burbeck, Steve. "Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC)."University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive
2. <https://msdn.microsoft.com>
3. <http://www.iis.net/>
4. KNOWLTON, CHRIS “IIS Media Services 3.0 Overview”, April 2010, <http://www.iis.net/learn/media/iis-media-services/iis-media-services-30-overview>
5. Zambelli, Alex, Media Technology Evangelist at Microsoft Corporation, “IIS Smooth Streaming Technical Overview”, March 2009