Czech Technical University in Prague
Faculty of Electrical Engineering

MASTER'S THESIS

Mobile iOS application for a non-profit TV

Prague, 2014 **Author: Mikhail Sukhotin**

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used. I have no objection to usage of this work in compliance with the act §60 Zakon c. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on _____ _____

signature

Acknowledgement

I would like to thank my supervisor, Ing. Tomáš Vondra, for his help and support with this thesis. I would also like to thank Mr. John Honner, founder and executive director of the Czech-American TV, for his cooperation and consultations.

Abstract

The main goal of this work is to design and implement a mobile iOS application for interacting with services of the american non-profit organization Czech-American TV. The application is able to play video and audio streams, interact with points of interest and provide an entertainment part in the form of quizzes about the culture of the Czech Republic. The mobile application is accompanied by a web application, which is implemented using Ruby on Rails framework. The mobile application communicates with server application using REST API through HTTP protocol. Web application supports HTTP Live Streaming and streams video and audio content.

Contents

Chapter 1

Introduction

Czech-American TV is a non-profit charitable organization that have a mission to support Czech cultural and educational programs via television and internet. The main program themes are czech language classes, czech kitchen, traditions, reports from Czech Republic, famous Czechs etc. Currently Czech-American TV broadcast across the USA and worldwide via internet on website `http://www.catvusa.com` and via cable TV every Wednesday in 60 U.S. cities. However nowadays usage of mobile devices intensively grows and mobile applications are now an integral part of almost every business, irrespective of their size and industry.

1.1 Problem description

The main goal of this project is to develop a software product which contains a client mobile iOS application and a server side web application for interacting with content of the Czech-American TV service. The service content is various and is regularly changed and updated. It is the reason why it is necessary to implement a web server, which processes requests from the client application and sends responses with an appropriate data. Receiving and playing video and audio streams is the main feature of the developing application. Video streaming in the Czech-American TV is currently realized by using the custom scripts, that send data directly to JWPlayer, responsible for playing a content. Unfortunately, this way has its problems with the implementation for the iOS platform, therefore it is necessary to work it over.

1.2 Functional requirements

Functional requirements specify the behaviors the product will exhibit under specific conditions. They describe what the developers must implement to enable users to accomplish their tasks.

1.2.1 Video streaming

A mobile application is required to display a list of available video categories (for now there are main broadcast, czech language classes, czech kitchen) and a list of all available for a current user videos in each category including archive (if available). All lists need to be downloaded and cached from a web server and it is required to be configurable through a web application.

After tapping on the video item application selected video stream starts. The application should support streams of different quality and should be able to automatically switch between the streams in case the available bandwidth changes. The next requirements are to remember the last watched position and to be able to resume the streaming in the future.

1.2.2 Audio streaming

The mobile application should display list of available radio streams (currently folk and classical music). The list is required to be downloaded and cached from the web server. By tapping on the radio stream item application starts to provide audio stream. During a playback it shows a meta information, provided by the server (currently a band name and an official website link). The radio stream list and the meta information for each stream should be configurable through the web application.

1.2.3 Content limitation

The application needs to support a membership system, which is implemented on the Czech-American TV web service. For a service member there are no limits

on a content consumption, while an unregistered user will have the following constraints:

- Only the last broadcasting video and currently promoted videos are available
- The radio stream is available only 15 minutes per day
- Only the most recent quiz is available
- The navigation option on a map is disabled

1.2.4 Map

The mobile application is required to be able to display a map with marked points of interest (POI). The list of POI is downloaded and cached from the web server. The map should not be limited by any territory - it displays points of interest worldwidely. By tapping on the POI mark on the map, the application displays detailed information about this point. Also, the application is required to be able to navigate a user to the selected points of interest. All the detailed information about the points of interest should be available and configurable on the web server.

1.2.5 Quiz

The application needs to contain an interactive and entertaining content in the form of a quiz. The list of available for a current user quizzes including an archive should be downloaded and cached from the server. Quiz questions and answers are configurable on the web server.

1.2.6 Social sharing

The main requirements on the social sharing is that the application has a possibility to share the specific video, radio, quiz, POI and other content via email, SMS and the most popular social networks: Facebook, Twitter, Google Plus etc. Also, the application should contain links to Czech-American TV accounts in these social networks.

1.2.7 Notifications

The application is required to be able to notify a user about updates in an application content.

1.2.8 Server side web application

The server side application should be available through web. It should grant access to the content configuration only after an authentication process. It is necessary for the web application to have a convenient way to configure all the content information: available video streams, a list of promoted videos, available radio streams, points of interest, quizzes.

1.3 Non-functional requirements

Non-functional requirements might specify not what the system does, but rather how well it does those things. They could describe important characteristics or properties of the system. These include the system's availability, usability, security, performance, and many other characteristics.

1.3.1 Platform

The client application needs to run on the iOS platform, and support all iOS mobile devices with different screen sizes and resolutions:

Device family	Supported screen sizes	Supported resolutions
iPhone/iPod	3.5", 4", 4.7", 5.5"	320x480, 320x568, 375x667, 414x736
iPad	7.9" and 9.7"	1024x768 and 2048x1536

The application is required to support both the landscape and the portrait device orientations for the iPad family. For the iPhone/iPad family of devices the main de-

vice orientation is the portrait with the exception of a video player, which supports all the possible orientations.

1.4 Chosen technologies

There are a lot of different tools and technologies that can be used in the implemention the project design into real world. I tried to choose instruments that are not only help me to fulfil the requirements of this project, but also will be valuable for me in the long term. For instance, Ruby on Rails framework was released not so long ago and now it is very popular and powerful tool for creating web applications.

1.4.1 Platform

The mobile iOS application is implemented in Objective-C language with the Cocoa Touch framework. According to the official statistics from Apple, measured by the App Store during a 7-day period ending November 24, 2014 there are 60% of devices are using iOS 8, 35% of devices are using iOS 7 and 5% of devices are using earlier versions [2]. Therefore the application implementation is directed on the support of iOS 7 and iOS 8 versions, but also has a backward compatibility with iOS 6.

A resolution and a screen size support are implemented using a new feature - Auto Layout [3]. The Auto Layout is a system that lets lay out application user interface by creating a mathematical description of the relationships between the elements. Developer defines these relationships in terms of constraints either on individual elements, or between sets of elements. Using Auto Layout, developer can create a dynamic and versatile interface that responds appropriately to changes in screen size, device orientation, and localization.

1.4.2 Video streaming

The video and audio streaming on the iOS platform can be implemented using HTTP Live Streaming technology [4]. HTTP Live Streaming lets developer send

audio and video over HTTP from an ordinary web server for playback on different kind of devices. On the server side there are scripts for creating necessary streaming files. In the client iOS application interaction with streaming content will be able through standard MPMoviePlayerController component from MediaPlayer framework. MPMoviePlayerController manages the playback of a movie from a file or a network stream.

1.4.3 Content limitation

The most convenient and Apple style implementation of the content limitation is the Apple payment system In-App Purchase [7]. In-App Purchase allows developers to sell a variety of items directly within its free or paid applications, including the premium content, virtual goods, and subscriptions. But just like applications selling on the App Store, developer receives 70% of the purchase price. It means 30% of a donation goes to Apple and this is the first reason to decline this kind of realization. The second reason is that we can't grant access to the users, who made donations through the website, because its not connected with the Apple in-app purchase system.

Fortunately, the Czech-American TV introduces new membership program for their service with a registration system. Application communicates with the Czech-American TV membership realization through its application server. The content limitation is implemented mostly on the server side and based on the data, obtained from user.

The radio stream limitation is realized using the standard functionality of NSUserDefaults class from the Foundation Framework. The NSUserDefaults class provides a programmatic interface for interacting with the defaults system. The defaults system allows an application to customize its behavior to match a user's preferences. The application in an unregistered mode will measure the time, which user spends on the radio streaming and when it exceeds 15 minutes its marked by a specific value in the application defaults.

1.4.4 Map

The map is realized using the Map Kit framework [8], which provides an interface for embedding maps directly into application views. This framework also provides

support for annotating the map, adding overlays, and performing reverse geocoding lookups to determine placemark information for a given map coordinate.

Routing in the application is implemented using the build-in function openMapsWithItems:launchOptions: in the class MKMapItem from the MapKit framework. This redirects user to the iOS Maps application where the user can use available routing features.

1.4.5 Social sharing

For the social sharing application UIActivityViewController from the UIKit framework is used [9]. The UIActivityViewController class is a standard view controller that developer can use to offer various services from its application. The system provides several standard services, such as copying items to the pasteboard, posting content to social media sites, sending items via email or SMS, and more. The application can also define the custom services.

Links to the Czech-American TV accounts in social networks is implemented using the basic functionality for opening URLs. For some services there are predefined url schemes. For instance, Facebook has `fb://`, Twitter has `twitter://`, Google Plus has `gplus://` etc. If there is a corresponding application installed, it is directly opened without using a browser.

1.4.6 Notifications

User notifications are implemented using local and push notifications. Local notifications and push notifications are ways for an application that isn't running in the foreground to let its users know it has an information for them. The information could be a message, an impending calendar event, or a new data on a remote server. When presented by the operating system, the local and push notifications look and sound the same. They can display an alert message or they can badge the application icon. They can also play a sound when the alert or badge number is shown.

The Apple Push Notification service (APNs for short) [13] is the centerpiece of the push notifications feature. It is a robust and highly efficient service for propagating information to iOS and OS X devices. Each device establishes an accredited and encrypted IP connection with the service and receives notifications over this per-

sistent connection. If a notification for an application arrives when that application is not running, the device alerts the user that the application has data waiting for it.

1.4.7 Serverside web application

Communication with the web server is based on JSON messages exchange. JSON is chosen since it is an open standard format that uses human-readable text. It is a independent and compact language.

The web server is implemented using the Ruby on Rails Framework [10]. It is an open source web application framework which runs on the Ruby programming language. It is a full-stack framework: it allows creating pages and applications that gather information from the web server, talk to or query the database, and render templates out of the box. As a result, Rails features a routing system that is independent of the web server.

For working with database during the development stage SQLite is used [11]. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is free and there is almost no need in configuration so it's perfect for the development. However due to the Ruby on Rails flexibility and easy customization, SQLite can be replaced with the another database management system by changing one configuration file.

Chapter 2

Client iOS application

2.1 Application design

Application design is the most important part of the development process. Designing, prototyping and wireframe creation can help to reveal possible weaknesses in the project requirements. Also, it can prevent a lot of future implementation errors, appearing due to developers misunderstanding of the project requirements. UML diagrams and wireframe creation were chosen for the application design in this project.

2.1.1 UML diagrams

UML (Unified Modeling Language) is a standardised modelling language enabling to specify, visualize, construct and document artifacts of a software system [14]. During the design phase of the project development several UML diagrams were created.

Use-case diagram

Use-case diagrams describe functionality of the system in terms of actors, goals and dependencies among the use cases. In a project client side there is only one actor - a mobile application. The mobile application provides four main functions: watching video broadcasts, listening to the radio, looking for the points of interest

on the map and taking the quizzes on different subjects (Figure 2.1). Each of these functions can be extended with function "share with friends". For instance, user can share currently watched broadcasts with his friends through a social network. Moreover user can send information about some point on the map to his friend via email. Also, the mobile application is required to provide turn-by-turn navigation directions to selected point of interest. That feature is delegated to the native Apple iOS application "Maps".

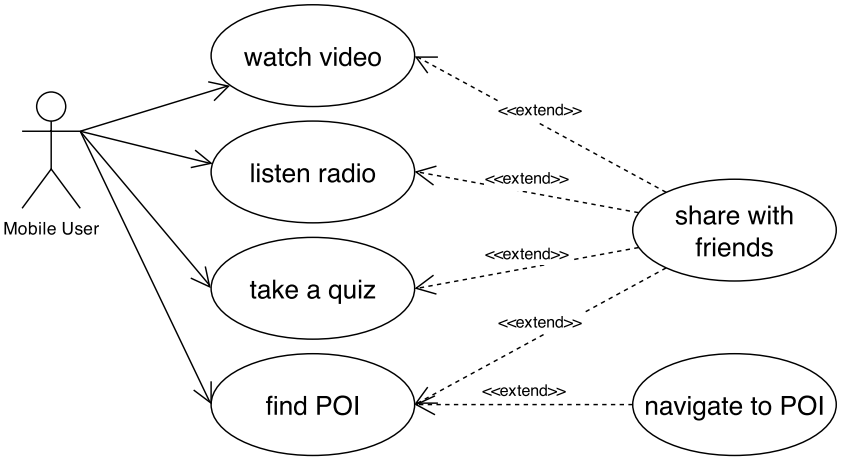

Figure 2.1: Use-case diagram.

Class diagram

Class diagrams represent system classes, attributes and relationships among the classes. The class diagram for this project is divided into two parts: the class diagram for the controllers (Figure 2.2) and the class diagram for the model (Figure 2.3). All controllers in the Cocoa framework are subclasses of the UIViewController class. Also, there is a very useful standard class UITableViewController, which is very helpful for working with tables. In this project, there is one root subclass of UITableViewController which is responsible for working with local data storage - UICoreDataTableViewController. All communication with the database and changes notifications are managed by this class. Some classes have the composition link to other "detailed" controllers. This kind of link in iOS development represented by the hierarchical navigation style where user navigates by making one choice per screen until he reaches his destination.

Sequence diagram

Sequences diagrams represent communication between objects in terms of a sequence of messages. Requesting the list of the data (Figure 2.4) is the most often operation in this project. Every controller in the application communicates with the local application cache in the first place. This behaviour has two big advantages. The first one is that the application is functional in the offline mode (except playing streams and browsing the map). The second one is that this increases application interface feedback to a user - he gets cached data almost instantly. At the same time, the application sends asynchronous request for data to the application server. If and only if there are any changes, the application updates the local cache and

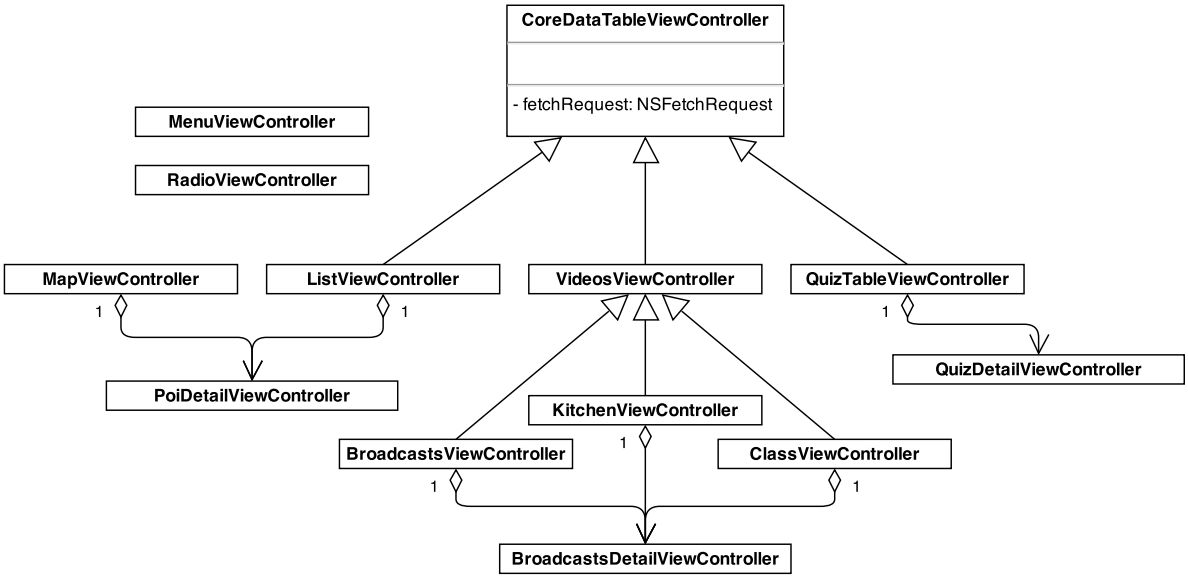

Figure 2.2: Controller's class diagram.

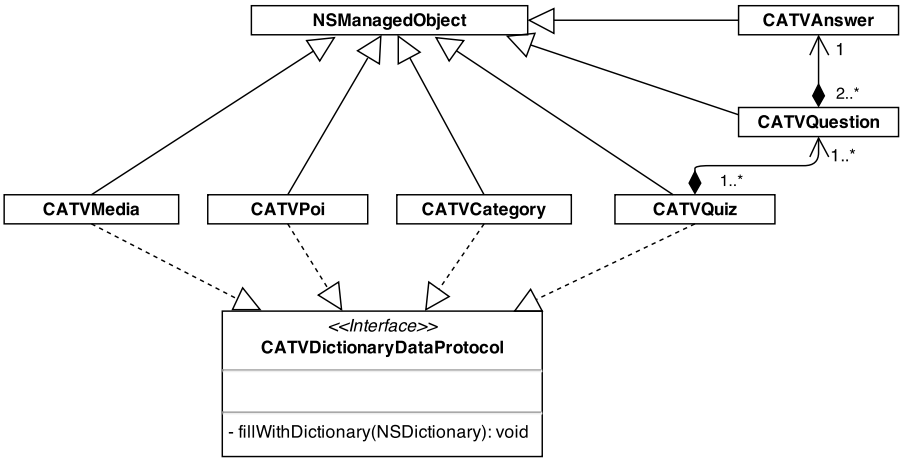

Figure 2.3: Model class diagram.

automatically notifies all the active corresponding controllers.

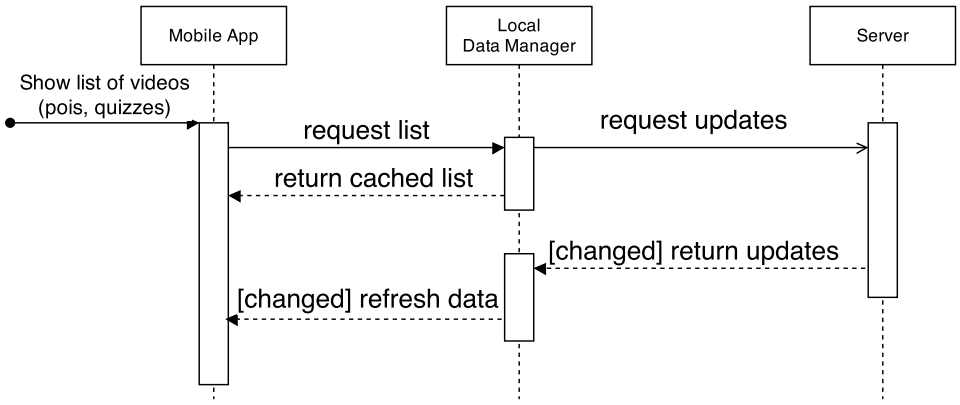

Figure 2.4: "Show list" sequence diagram.

The second diagram represents the sequence of actions which is performed when a user starts playback of the media stream. First of all, the client application asks the application server for the link to the stream playlist file. If the application server already has cached url to requested resource - it sends it back to the user. Otherwise, the application server sends a request to the content server which is responsible for the streaming media content. The content server generates a temporary url link to the requested resource and sends it back to the application server, which passes it next to the client application. Afterwards, the mobile application communicates directly with the content server.

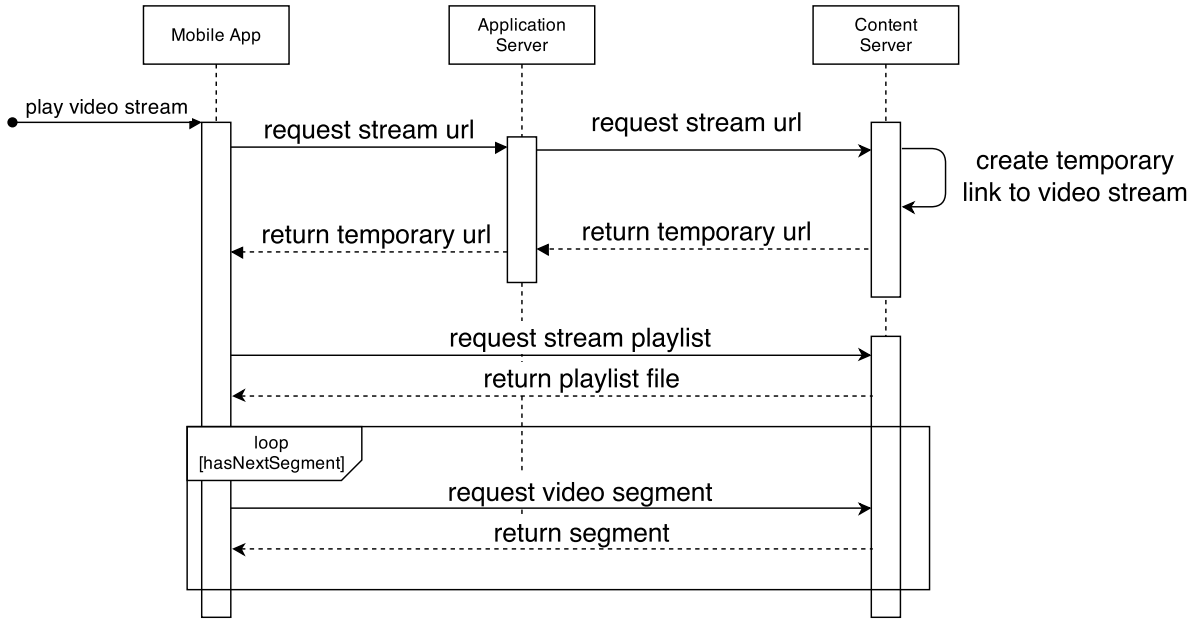

Figure 2.5: "Play stream" sequence diagram.

2.1.2 Application prototypes. Initial design

The next phase of the application design stage is a creation of the application prototypes. These wireframes on the one hand show navigation inside the application and on the other hand determine the way how this application looks like.

The first wireframe (Figure 2.6) shows the main navigation element of the application - the application menu, which is presented by the navigation drawer design pattern. The navigation drawer is a panel that transitions in from the left edge of the screen and displays the application's main navigation options [18]. The big advantage of applying of this approach on mobile devices is that navigation in the application is moved out from the main screen and there is more free space for displaying the content.

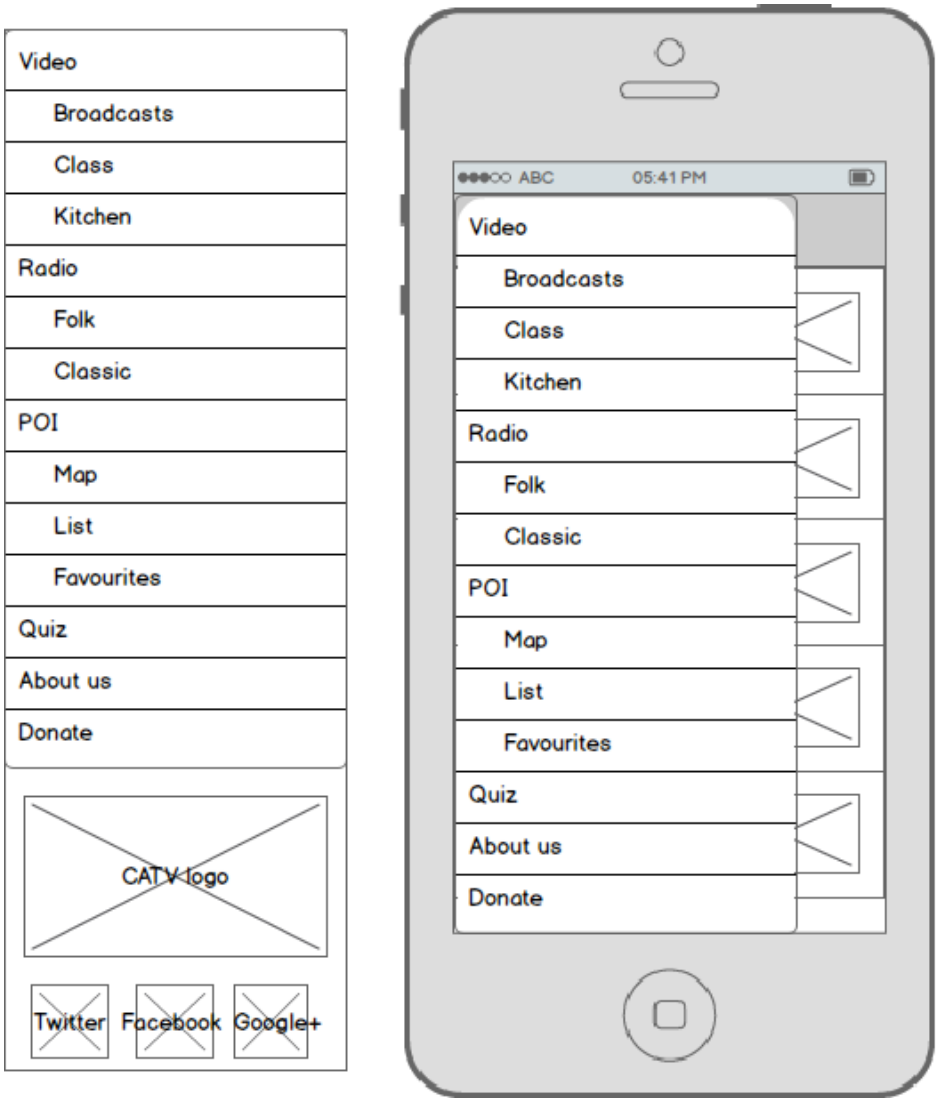

Figure 2.6: Drawer menu

The navigation menu has a hierarchical structure and it is divided on several blocks: video streams, radio streams, working with points of interest, quizzes, information about organization, instructions for donating to the organization and block with image links to the website of the organization and to the accounts on three popular social networks. Tapping on any menu item replaces application central view controller with the selected controller and automatically hides the side navigation menu.

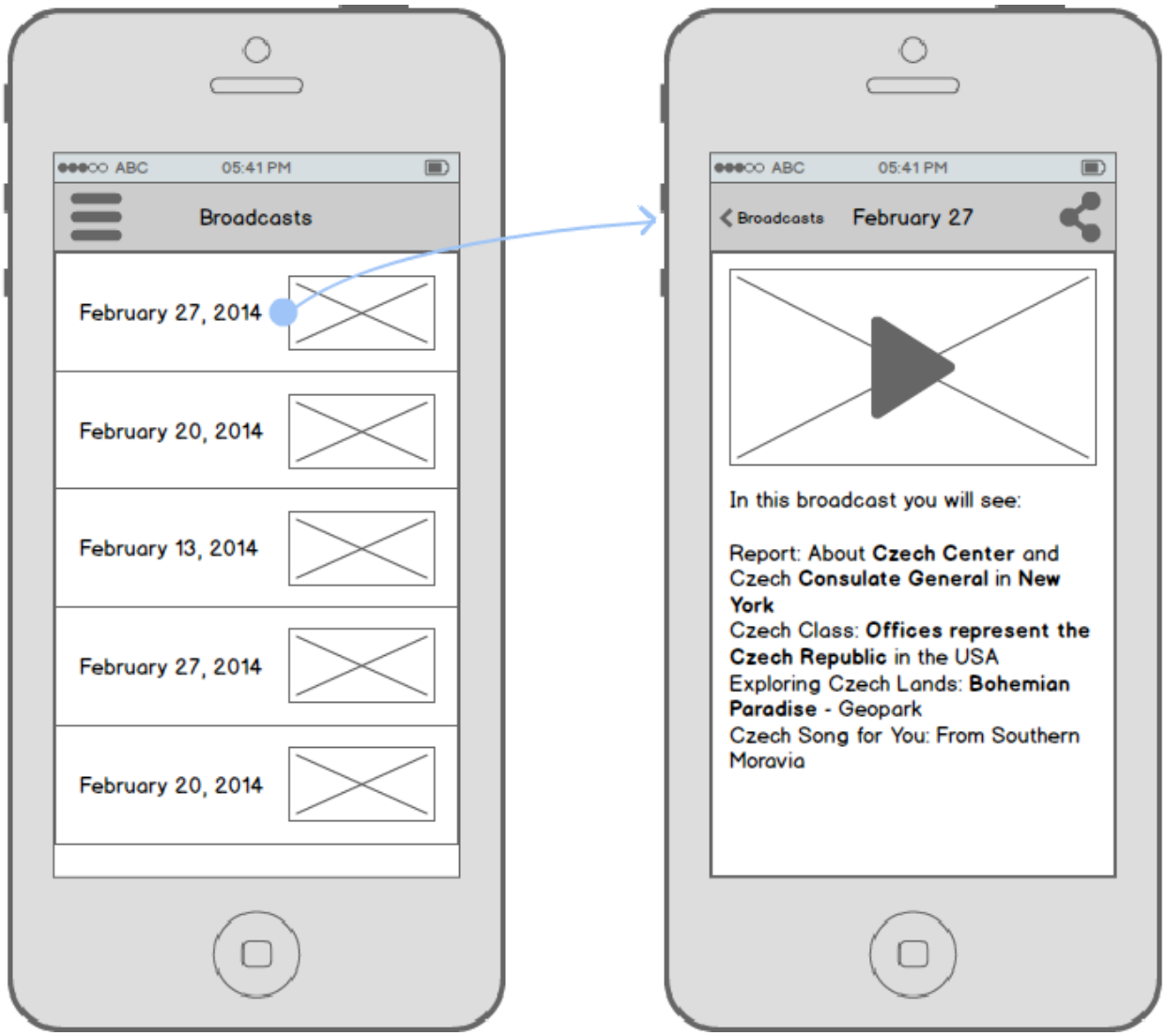

Figure 2.7: Resource list and detail controller.

The next wireframe demonstrates displaying the list of the data in the application (Figure 2.7). Table view is a common way for representing a data in mobile devices. Tapping on cell replaces the list controller with the detail controller which has more particular information about the data item. There is a table with recent broadcasts

and broadcast's detail controller here with detailed information about its content with the button for playing video stream.

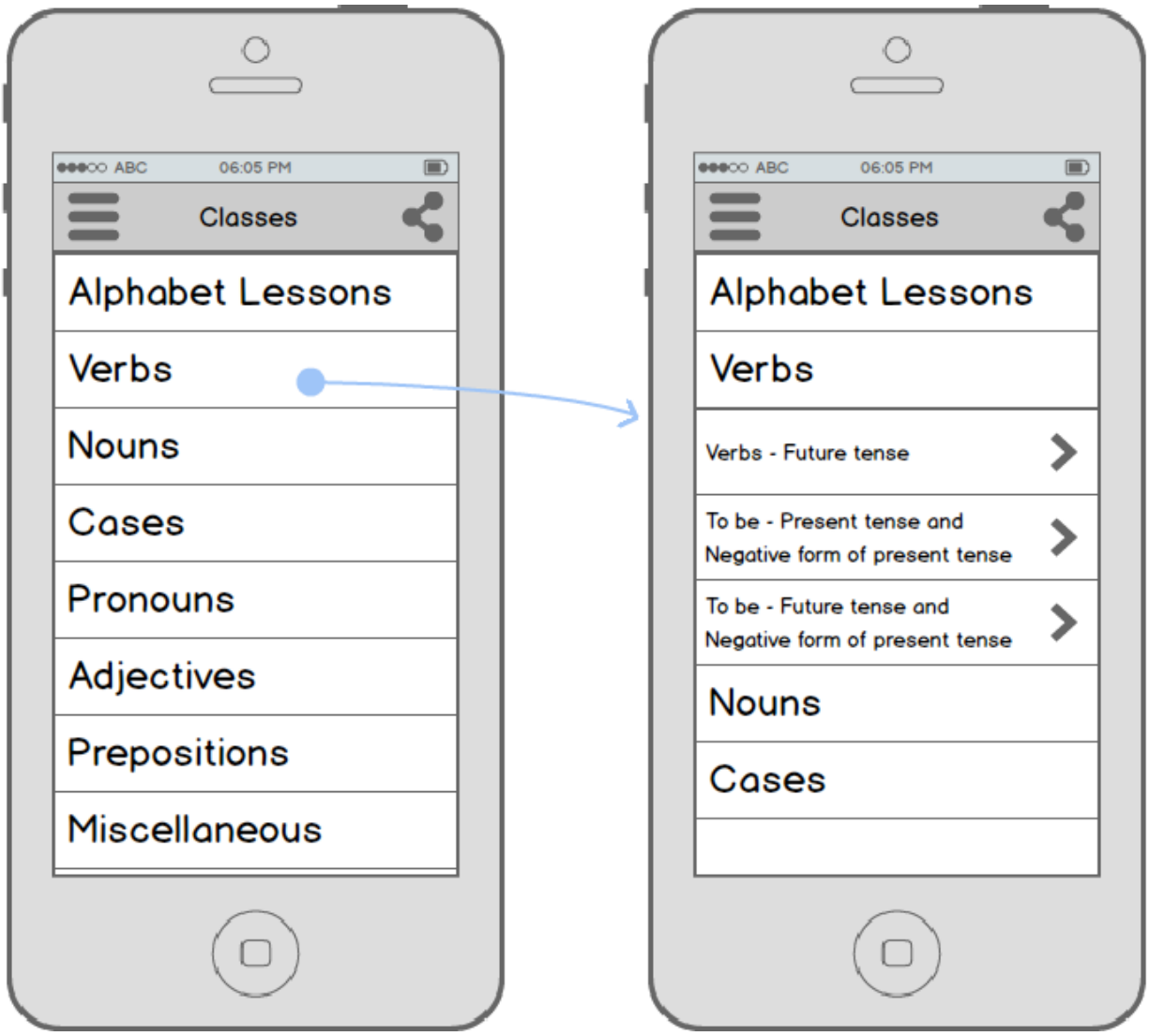

Figure 2.8: Another kind of lists representation

Displaying the list of czech language classes videos is slightly different from others due to its hierarchical structure (Figure 2.8). The classes on the original website are divided by several categories such as: lessons, verbs, nouns etc. At the beginning, the list of all these categories is displayed on the screen. When a user tapped on some category, it expands and shows a corresponding video lessons under the selected category. This kind of list representation helps to organize data without necessity to include another transition controller.

Currently, there are only two radio broadcasts in the Czech-American TV, so there is no need in list representation of radio resources. The process of selecting radio

in the side menu directly opens the detail controller with the corresponding radio information and the button for starting playback (Figure 2.9).

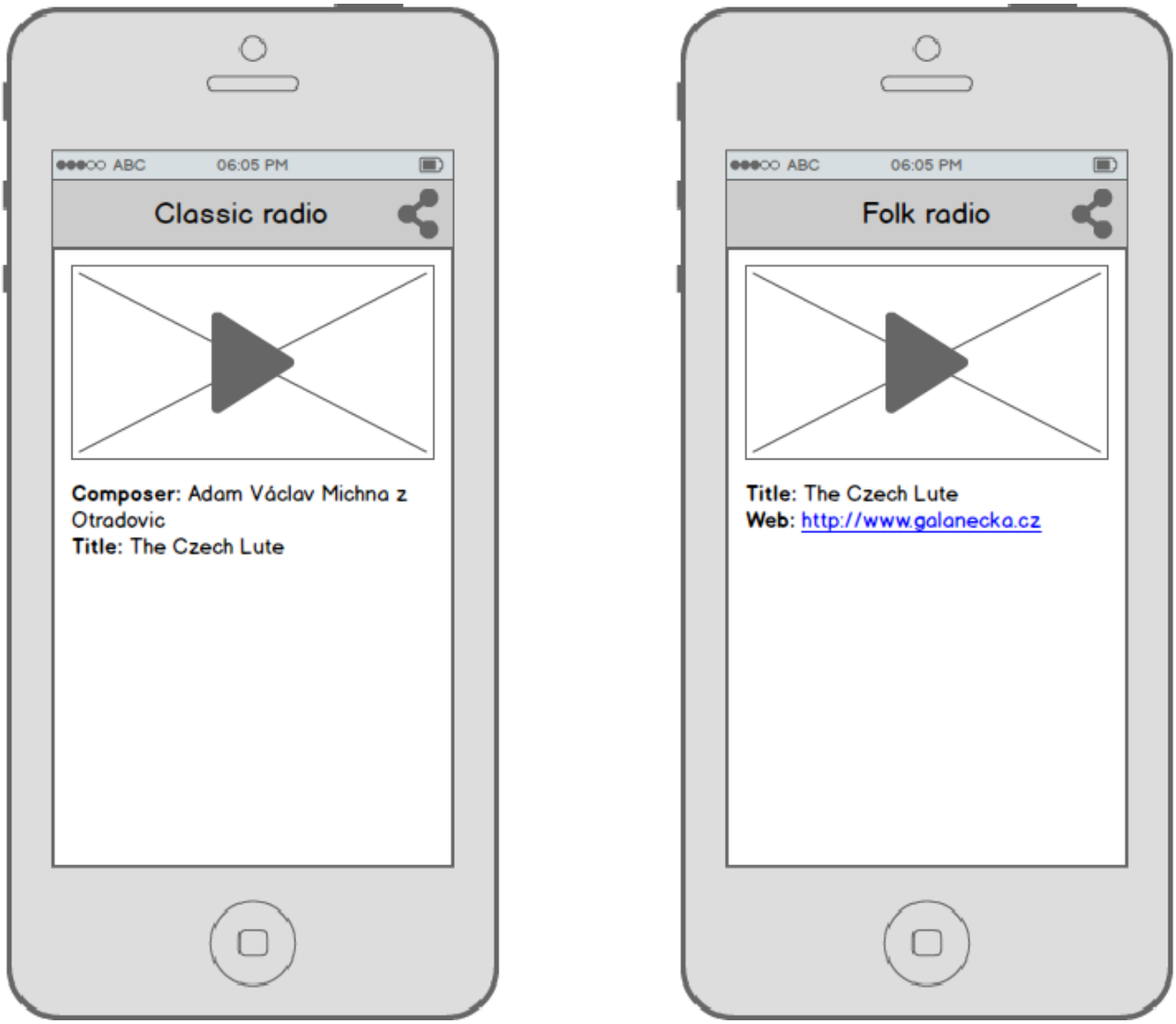

Figure 2.9: Radio controllers

Working with points of interest is realized using two controllers: MapViewController and ListViewController (Figure 2.10). ListViewController looks the same way as other table controllers. It displays all points of interest as a list. Tapping on a table cell opens the controller with a detailed information about the point of interest. The only difference is that this controller allows full text search through the points of interest. MapViewController is represented by the map view. A user can scale, move and rotate the map the same way as in other applications interacting with maps. All the points of interest are located on the map as a place marks. Selecting a mark displays its name and button, tapping on which leads to a more detailed view of the point of interest.

The detailed controller displays an information about address, website, phone number, opening hours etc. Also, a user can ask for turn-by-turn directions to the selected point of interests. The Maps is the preinstalled Apple application providing turn-by-turn directions to other applications, which don't have its own navigation services.

Figure 2.10: POI controllers

The quizzes controller is another representation of the table view controller (Figure 2.11). It displays a name of quiz, creation date and a current user's score for each quiz. Tapping on a cell opens the detailed controller with the last unanswered question (or with the first question in case of first opening). Selection of any answer is immediately changes question. User knows his score only at the end of the quiz. Also, there is a sharing button for posting the result to the social networks.

2.1.3 Application prototypes. Redesign

Application navigation in the new design based on the tab bar interface (Figure 2.12). Tab bars are used to organize information at the application level and represents the top level of the application hierarchy. Tab bars are widely used by Apple itself in their own applications so users already used to them. The main advantage of them in the user interaction sense is that user immediately see his location in the application and all other possibilities there he can go from here. Also, tab bar is located at the bottom of the mobile screen so it can be easily reached by a user no matter how large the screen is.

The application is divided on five top level sections each of them is represented by a single tab in the tab bar. These sections are video content, radio content, points of the interest on the map, quizzes and application information screen. The first tab contains video content that is organized with help of the UISegmentedControl at the top of the screen. That control is quite similar to the tab bar and displays all possible choices and highlight currently visible. Segmented control switches currently displayed video section between broadcast, kitchen, classes and favourites.

Information displayed differently on mobile phones and tablets. On the iPhones

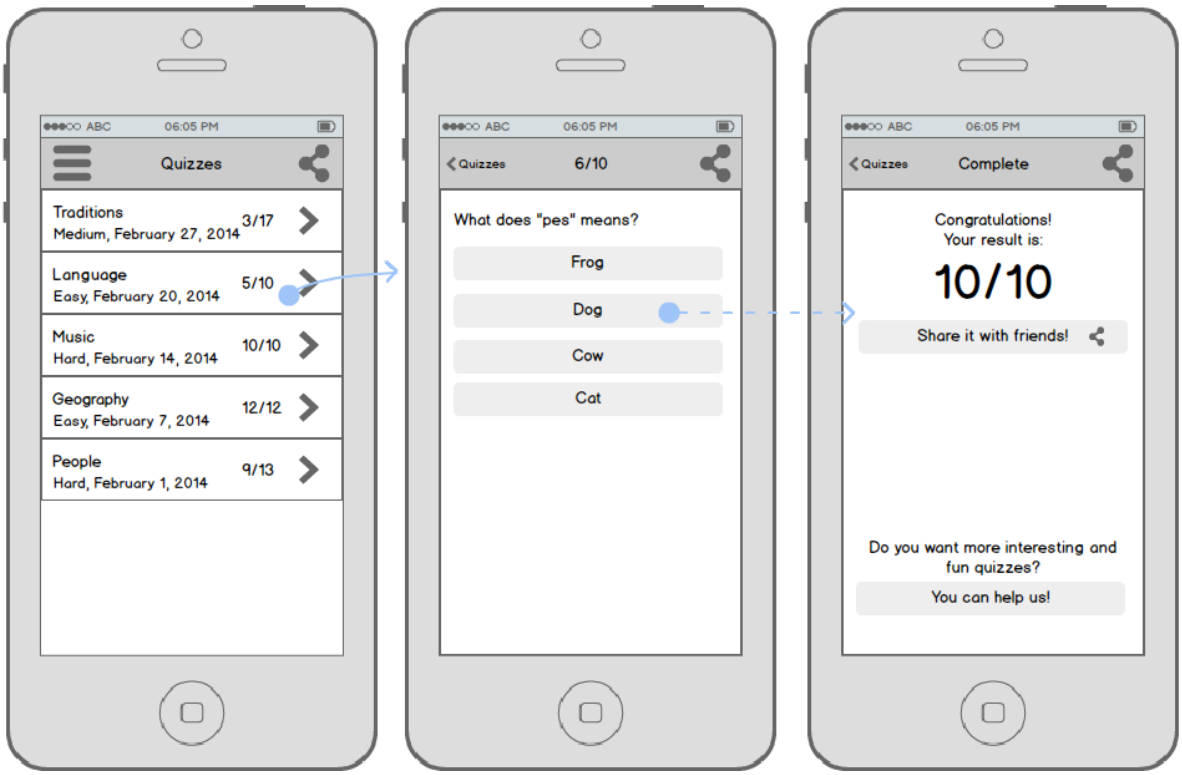

Figure 2.11: Quizzes controllers

content displayed in the table view, whereas on the iPads it is displayed as a grid. Tablets have much more space on the screen so application can also use horizontal direction for the content layout. Both views support searching through the content that can be reached by scrolling content down. Searching results filtered out in the same view as unfiltered content and all standard actions are also available.

New design use video detail screen only for the broadcast videos, because there are no additional information about videos in the kitchen and the classes categories. Instead of displaying detail, tap on the video view immediately shows available actions such as play, share and add to favourites. On the iPhone these actions displayed as a list of actions that appears from the bottom of the screen, when on the iPad this list is looks like context menu from the exact tap point. This difference is based on the distinction between two form factors: tablets and mobile phones. User mainly handle his mobile phone by one hand and operates with help of his

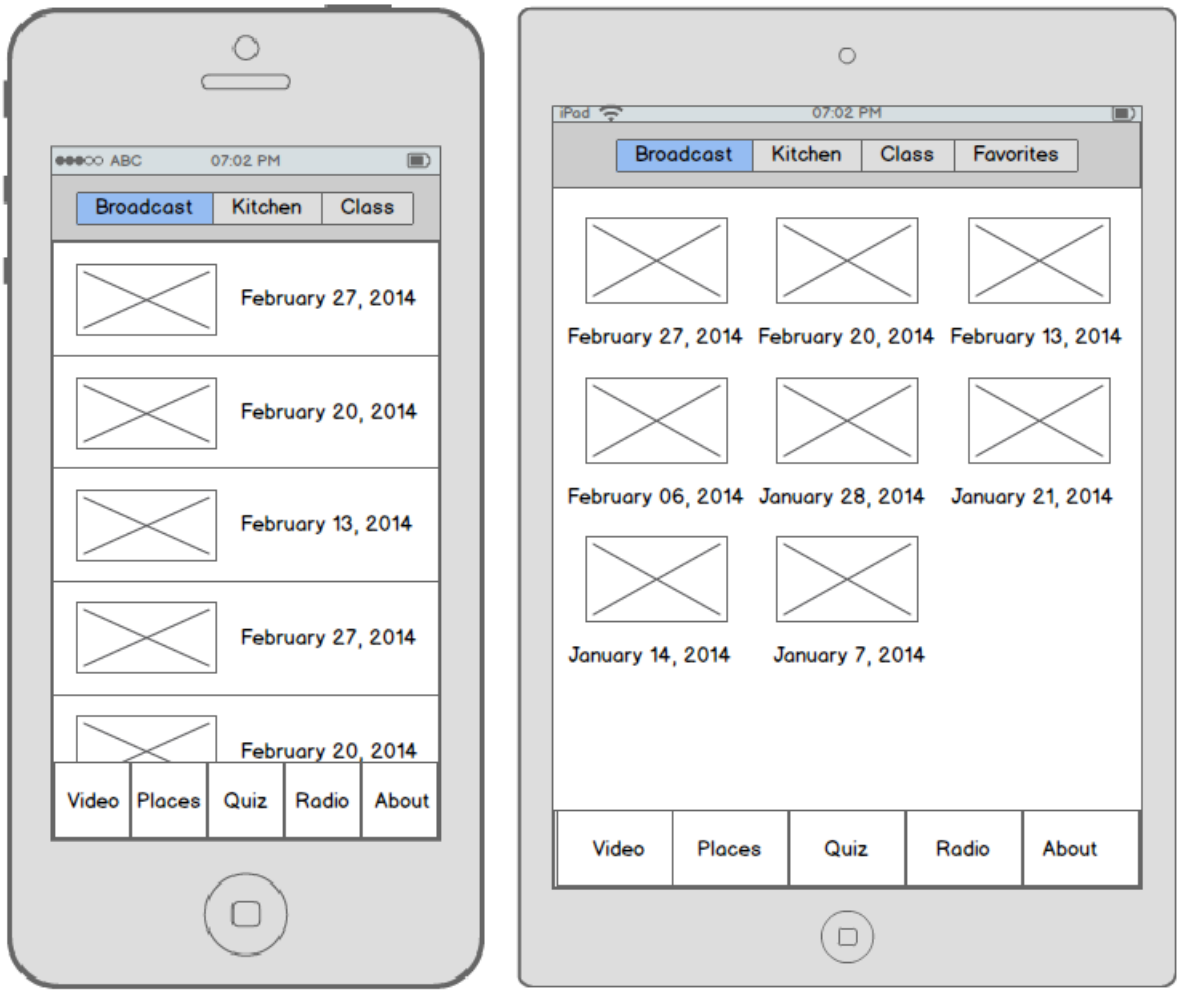

Figure 2.12: Redesign

thumb. Whereas on the tablets user handle the device by one hand and operate with it with help of the other hand.

Video detail view remains almost the same as in the initial design. Additionally, the chapters table were reorganized and supplemented with images. Also, there is a favourite icon at the top toolbar, with help of which user can mark videos and create his own list in the favourites tab.

Points of the interest on the iPhone are now displayed in the one combined tab that contains segmented control at the top of the screen, with help of which user can switch between the table view and the map view. On the iPad the table view and the map are displayed together. Search through the points of the interest and add to favourites action is also added for this section.

The separate tab dedicated to the About screen. It contains links to the Czech American TV website and to the donation page. Also, there are links to the most popular social networks where the organization has profiles.

2.2 Used technologies

On the client side technologies are chosen in favour of Apple products. Mainly because of these products are very powerful and efficient and there is no reason to choose alternatives.

2.2.1 Development environment

Xcode was chosen as the integrated development environment. It contains a suite of software development tools, released by Apple for developing software for OS X and iOS. Basically, there is only one alternative to Xcode - JetBrains' AppCode [12]. But the AppCode has a very poor support of the CoreData and storyboards which are heavily used in this project. The application was tested on an iOS simulator, that is also provided by Apple as a part of the Xcode tools. It behaves like a standard Mac application while simulating an iPhone or iPad environment. For this project simulator covers all required functionality, thus there is no need for real iOS device on the development stage. However, before production, testing on real devices is necessary because of possible bugs and performance problems that cannot be

caught during testing in the iOS simulator.

2.2.2 Design patterns

Design patterns are abstract solutions to common and well-known problems. Nowadays know patterns and know how to use them is almost mandatory for each software developer. The most important patterns that are used in this project are described in this chapter.

2.2.2.1 MVC

The Model/View/Controller (MVC) triad of classes is used to build user interfaces in Smalltalk-80. MVC consists of three kinds of objects. The Model is the application object, the View is its screen presentation, and the Controller defines the way the user interface reacts to user input. Before MVC, user interface designs tended to lump these objects together. MVC decouples them to increase flexibility and reuse [15].

MVC pattern is widely used in developing iOS applications. The benefits of adopting this pattern are numerous. Many objects in these applications tend to be more reusable, and their interfaces tend to be better defined. Applications having an MVC design are also more easily extensible than other applications. Moreover, many Cocoa technologies and architectures are based on MVC and require that your custom objects play one of the MVC roles [16].

Model objects encapsulate the data, specific to an application and define the logic. In this application, model is controlled by the Core Data framework and is represented by NSManagedObject subclasses: CATVPoi, CATVCategory, CATVMedia etc.

A view object is an object in an application that users can see. A view object knows how to draw itself and can respond to user actions. A major purpose of view objects is to display data from the application's model objects and to enable the editing of that data. In this application the view is represented by different subclasses of UIView class, mostly predefined by the built-in Cocoa framework.

A controller object acts as an intermediary between one or more of an application's view objects and one or more of its model objects. Controller objects are thus a conduit through which view objects learn about changes in model objects and vice

versa. Controller objects can also perform setup and coordinating tasks for an application and manage the life cycles of other objects. In the currently developing application, controllers are represented by different subclasses of the UIViewController class: MenuViewController, KitchenViewController etc. For instance, in the developed application there is the controller BroadcastViewController which uses UITableView as a view object to display an array of the CATVMedia model objects.

2.2.2.2 Singleton

Singleton pattern ensures a class that only has one instance and provides a global point of access to it. Developer obtains the global instance from a singleton class through a factory method. The class lazily creates its instance the first time it is requested and thereafter ensures that no other instance can be created [15] [17]. Several Cocoa framework classes are singletons. They include NSFileManager, NSWorkspace, and, in UIKit, UIApplication and UIAccelerometer. In this application a singleton object is used for the DataStore class which is a global point for accessing and manipulating with a data. It provides NSManagedObjectContext which is used by the Core Data framework classes and receives requests from a user to update and refresh an application data such as quiz questions, media objects, points of interest and others.

2.2.2.3 Drawer

Navigation drawer is a panel that transitions in from the left edge of the screen and displays the application main navigation options [18]. A user can bring the navigation drawer onto the screen by swiping from the left edge of the screen or by touching the application icon on the top bar. This is a relatively new design pattern, main goal of which is to replace common tabs bar or navigation bar menus in case of many menu items. In this application the navigation drawer pattern is implemented using the open-source project MMDrawerController under the MIT license. This library is designed to exclusively support side drawer navigation in a light-weight, focused approach while exposing the ability to provide custom animations for presenting and dismissing the drawer [19].

2.2.2.4 Delegate

Delegation is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object [20]. The delegating object

keeps a reference to the other object - the delegate - and at the appropriate time sends a message to it. The message informs the delegate of an event that the delegating object is about to handle or has just handled. The delegate may respond to the message by updating the appearance or state of itself or other objects in the application, and in some cases it can return a value that affects how an impending event is handled.

Delegation pattern is commonly used in Cocoa classes. One of the most using view classes UITableView requires implementation of the UITableViewDataSource delegate in which UITableView asks another object for number of the displaying rows and the view for each row. In this project there is a PoiDetailViewControllerDelegate which has method showPoiOnMap using which the main class delegates displaying point of interest to another class.

2.2.3 Core Data

The Core Data framework provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence. In this project Core Data represents a model part of the Model-View-Controller pattern. Core Data provides an infrastructure for change management and for saving objects to and retrieving them from storage. It can use SQLite as one of its persistent store types [21].

Important parts and classes of the Core Data framework, which are used in this project:

- Managed object model
- Managed object
- Managed object context
- Persistent store
- Fetched results controller

A managed object model is an instance of the NSManagedObjectModel class. It describes a schema, a collection of entities, that is shown in your application.

Managed objects are instances of the NSManagedObject class that represent instances of an entity. NSManagedObject is a generic class that implements all the basic behavior, required for a managed object.

An instance of NSManagedObjectContext represents a single "object space" or scratch pad in the application. Its primary responsibility is to manage a collection of managed objects. A single managed object instance exists in one and only one context, but multiple copies of an object can exist in different contexts.

Core Data provides three sorts of disk-based persistent store - XML, atomic, and SQLite - and an in-memory store. SQLite persistent store is used in this project.

NSFetchedResultsController is used for efficiently manage the results returned from a Core Data fetch request to provide a data for a UITableView object.

2.2.4 Auto Layout

Auto Layout technology was introduced at the Worldwide Developer Conference in June 2012 together with the 6th version of the iOS [3]. It was a first step that Apple did towards to mobile devices with various screen sizes (next ones were shift from skeuomorphism to a more simplified design in iOS 7 and size classes in iOS 8). Auto Layout describes location of the user interface elements in terms of constraints. In this project Auto Layout used in all table cells and in the About screen, thats why the application displayed correctly on all Apple devices.

Chapter 3

Server application

3.1 Application design

This chapter describes three most important problems in the server application design: in what way the data will be stored, how the client application will communicate with the server application and how to manage the server content.

3.1.1 Database model

Relational database was chosen for this project due to its popularity and installation simplicity. Also, there is a default and easy to use the Ruby on Rails class ActiveRecord, which represents the commonly used architectural pattern Active Record. Seven relational tables were created for this project (Figure 3.1). These tables can be divided on two main groups: categorized relations and quiz relations.

The first group is represented by three tables: categories, media and POIs. Video streams and radio streams are merge to one table because of theirs similarity. Each media object has one category and may have one subcategory. There are no plans in the Czech-American TV for supporting nested categorization so there is no need in recursive associations. POIs table contains columns for address and coordinates that may seem redundant, but geocoding services are still not so precise in many countries so human readable and correct address is still needed. The media table and the POIs table are in an one-to-many relationship with the category table.

The second group is represented by four tables: quizzes, questions, questions_quizzes

and answers. The quizzes table contains only one field (excluding indentifier) - name, which simply represents the quiz name. The questions table also contains only one text field for the question's text. These two tables are connected by a many-to-many relationship in table questions_quizzes, which contains primary keys of each member of the relationship. This relationship allows to share questions between different quizzes. The answers table among with the answer text field also contains a boolean flag that indicates wether it is a correct answer or not. Moving information about the correct answer to the answer table allows to create multiple answer questions. The answers table is in a one-to-many relationship with the questions table.

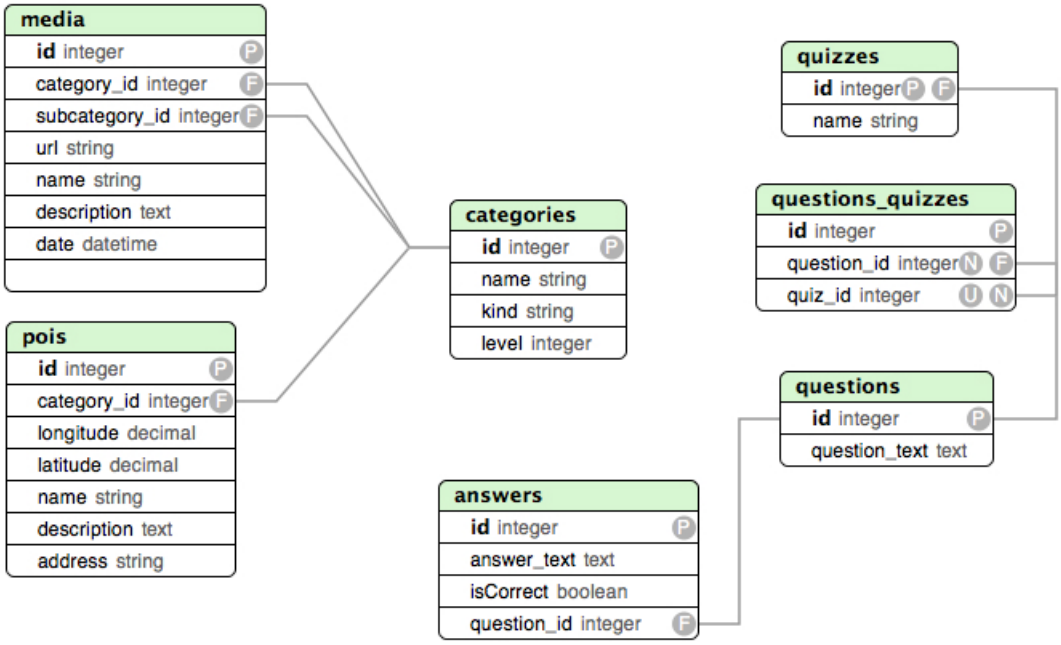

Figure 3.1: Database model

3.1.2 Client-server interaction

The server side contains two main parts: the application server with the RESTful web service and the content server with the web application for delivering it (Figure 3.2). Both applications are written using the Ruby on Rails framework. Currently used database is SQLite, but it also can be easily replaced by any other database management system like MySQL, PostgreSQL and others and moved out to a separate server (during development SQLite database situated on the same server as a web application).

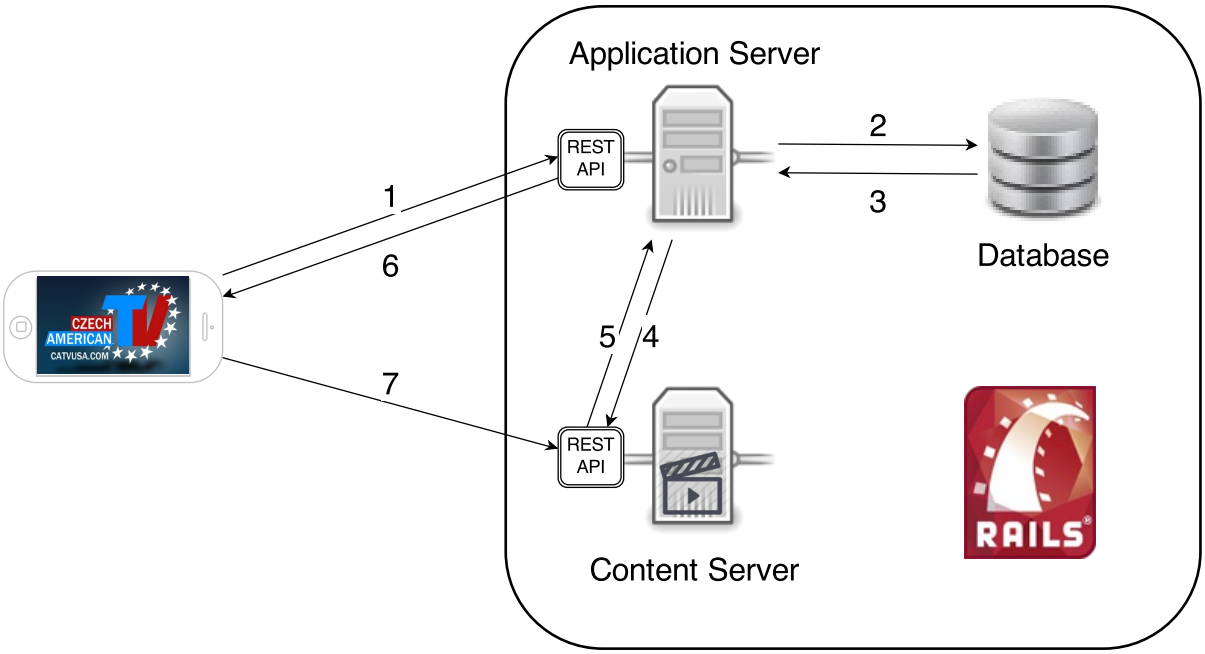

Figure 3.2: Client-server interaction scheme

The application server is managed by the Ruby on Rails RESTful web service, which is stateless and can be easily scalable to several instances behind a load balancer. The main goal of this server is to provide all the necessary information for the client application in a cross-platform format (json or xml) such as a list of available broadcasts, list of points of interest, questions and answers for quizzes etc.

All requests for a video stream, such as request for playlist (.m3u8) or specific media segment (.ts), are redirected by the application server to the content server. The content server web application generates url for a request and can be based on a user access permissions. The content server application generates url in the following format: `http://server_address/hls/random_string/video.m3u8`, where random_string is generated randomly and can be associated with access permissions of a requesting user. Also, this url has it's own time-to-live which can prevent from collecting permanent links to video streams by users.

3.1.3 Administration

The Czech-American TV regularly updates a video data such as broadcasts, czech language classes, kitchen videos etc. Currently, data changes and new posts are entering through a Joomla content management system. There is several reasons to make a separate management system for this project. Firstly, Joomla's system's

interface is pretty standard and general so there is no individual interface solutions for the current project. The second reason is that currently the Czech-American TV doesn't support points of interest and quizzes. And the third reason is that implementing content management system on the Ruby on Rails technology stack is quite easy for development and support and there is a chance, that the Czech-American TV will migrate on this platform in the future.

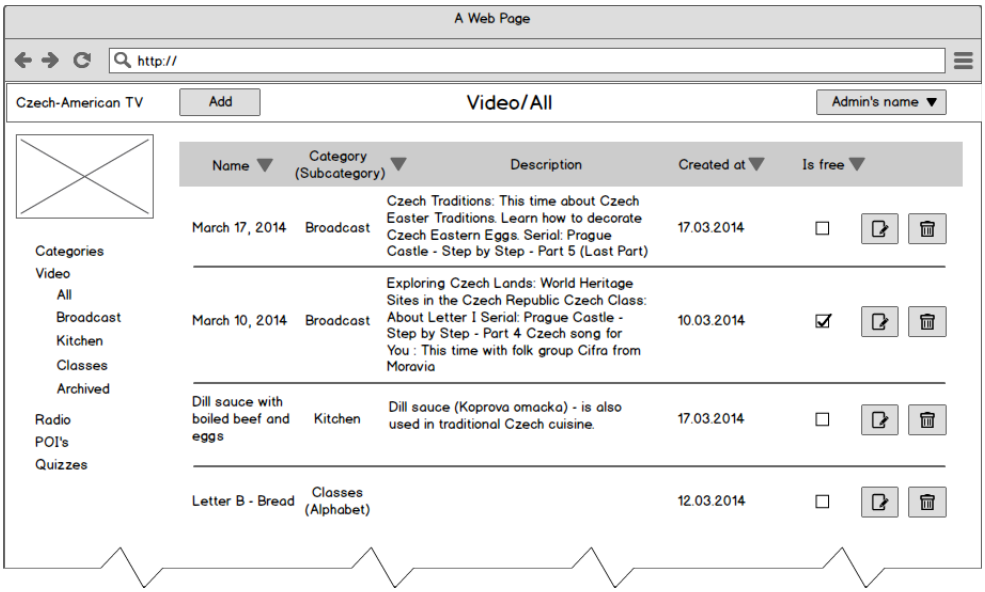

Figure 3.3: Video administration wireframe

The new content management system contains three main parts. The top bar displays the logo of the organization and a currently logged-in administrator and can have a set of an action buttons, which is depends on a context. The left bar represents menu of the CMS. It is divided on several sections: categories, video, radio, POI's and quizzes. Some sections can have a subsections which are simply different filters for the current section. For instance, the video section can have a Kitchen subsection which displays only kitchen related videos. Also, each section has an "archived" subsection which contains all deleted entities. And the third one part is a content part.

The first administration wireframe represents the video section (Figure 3.3). All videos are displayed as a table and each row can be edited or removed. Removing entity moves it to the archived subsection. Also, there is a checkbox "Is free" which indicates wether this video is available for an unregistered user or not. In the top bar of the screen there is an action button "Add" which is used for creating a new table item. The detailed view of the video entity is represented on the Figure 3.4. Category and subcategory fields are represented by the combo boxes which are eliminate the possibility of misprinting.

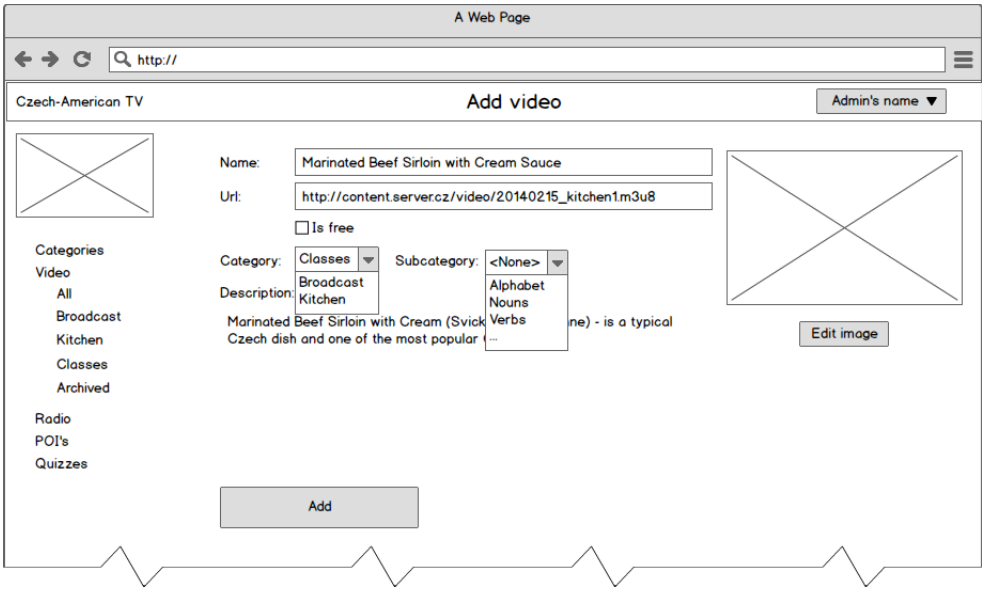

Figure 3.4: Detailed video view

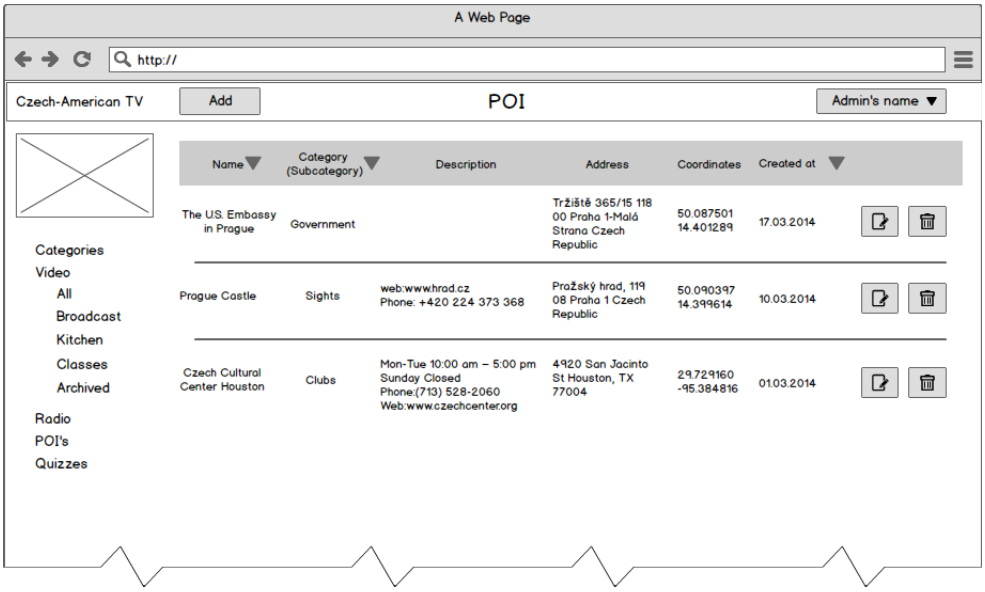

Figure 3.5: Points of interest

The table of points of interest looks almost the same way as the video table (Figure 3.5). The main difference is in the detailed view of the point of interest (Figure 3.6). There are several simple input elements such as the name field, the description field and the category combo box. And there is a one complex location input view. It contains address text field, coordinates text field and a map view. All three of these inputs are linked and can fill each other.

For instance, an administrator can input an address of the point of interest and using the "Get coordinates" button receive coordinates from a geocoding service and

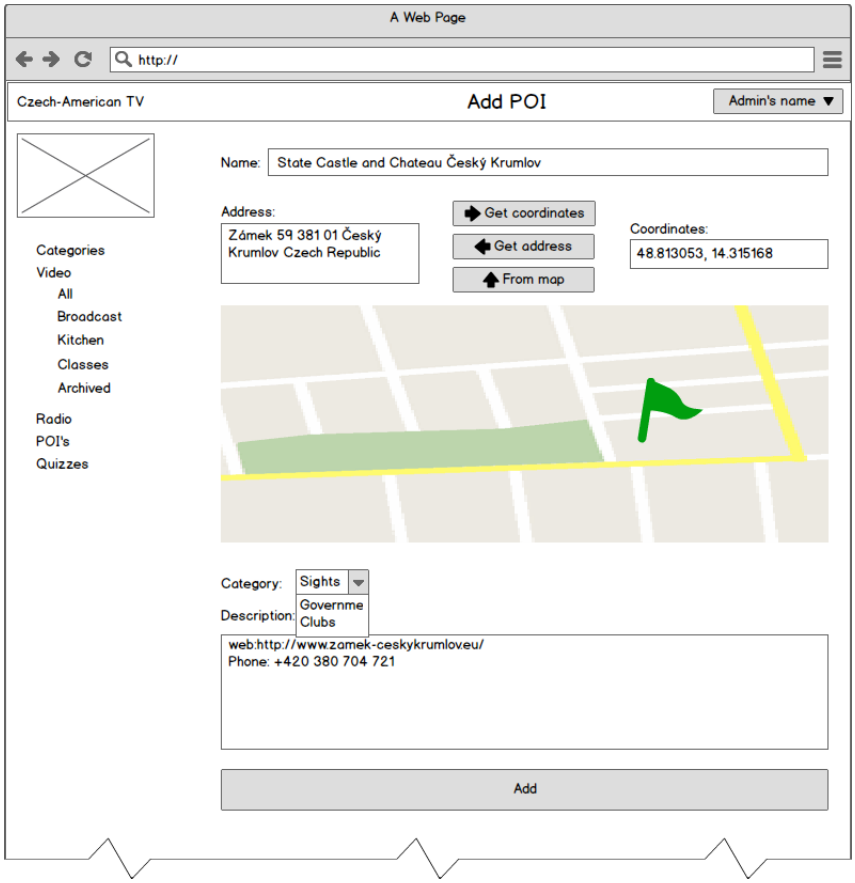

Figure 3.6: Detailed POI view

displays this point on the map view. Using the "Get address" button and the coordinates text field an administrator can receive an address from a reverse geocoding service and also displays it on the map. And there is the third option - locate the point on the map and get an address and coordinates. An administrator can edit received location information due to an imperfection of the geocoding services in different countries and, sometimes, not very clean and pretty address detection. The address can be easily edited in its text field and coordinates can be changed either through the coordinates text field or directly on the map.

The table of quizzes also looks like all other common tables (Figure 3.7). Creating a new quiz or editing an existing quiz redirects administrator to the detailed quiz view (Figure 3.8). Here there is a text field for the quiz name and the list of questions. Questions can be created (button "Create new question") or can be chosen from already existing questions from the database (button "Select from DB", Figure 3.9). Each question row displays question's text, answer alternatives (correct ones selected with a different color) and action buttons. Each question can be edited or removed from this quiz. Also, there is a buttons for ordering questions.

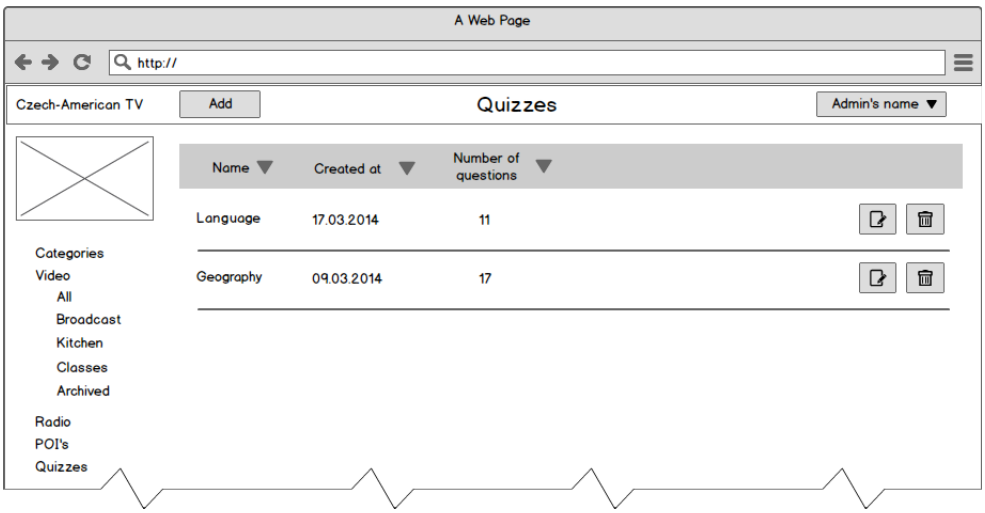

Figure 3.7: All quizzes

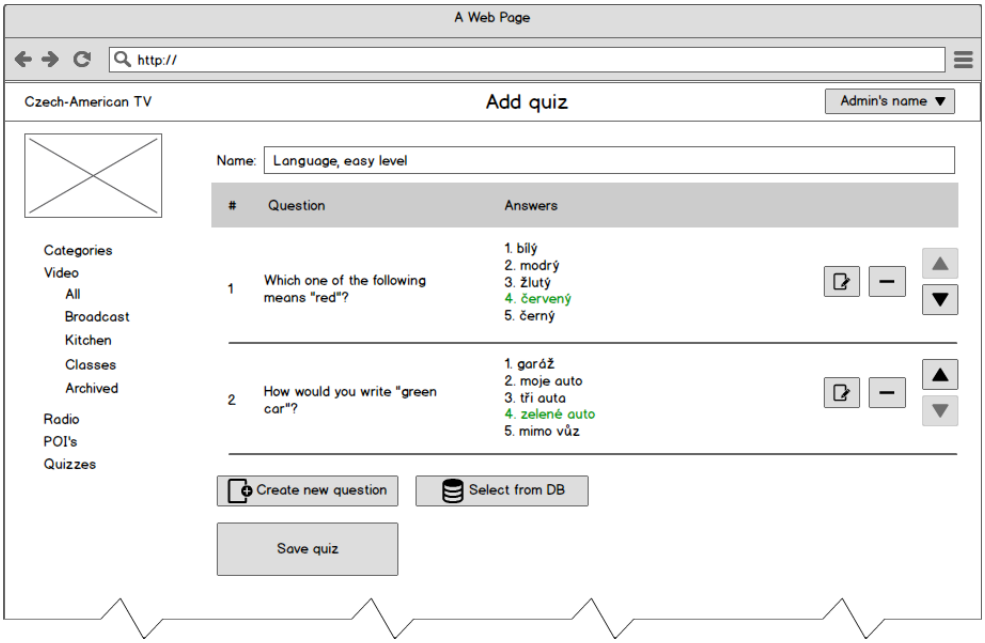

Figure 3.8: Detailed quiz view

3.2 Used technologies

Modern and powerful technologies are used in this project. RESTful web service that based on the Ruby on Rails framework that can stream video content through the HTTP protocol is very powerful and efficient combination of technologies for this project.

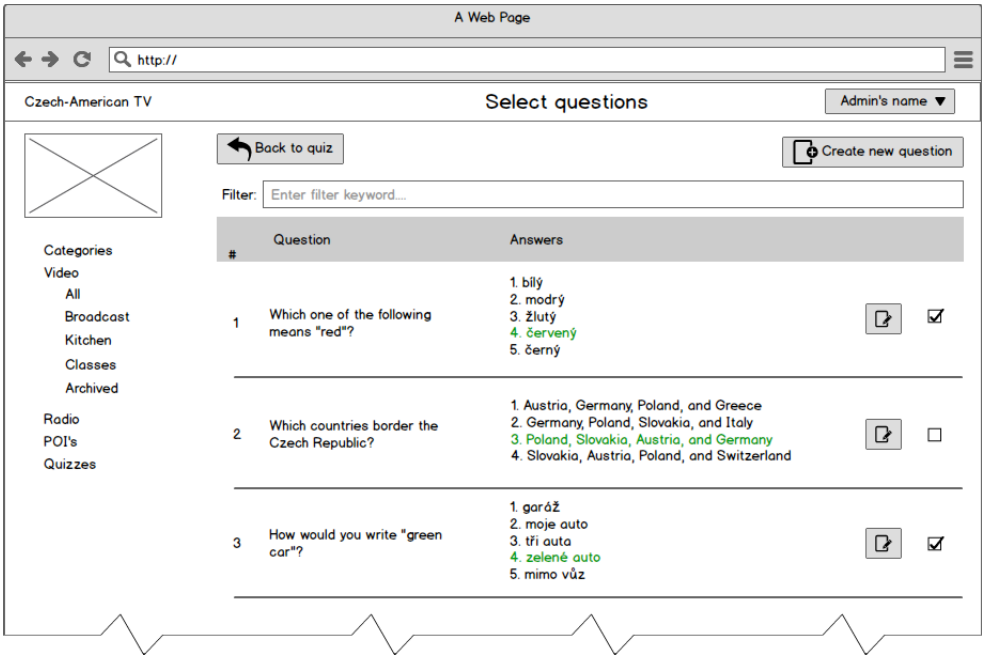

Figure 3.9: Questions management

3.2.1 Ruby on Rails

Ruby on Rails [10] is an open source full-stack web application framework, which means that all layers are built to work seamlessly together. Developer can use a single language (in this case Ruby) from view templates to a business logic. Ruby on Rails provide the library WEBrick for creating simple HTTP web server service, which is used in this project on the development stage. However using different gems developer can easily deploys a Ruby on Rails application on different others web servers such as Apache, nginx, Glassfish, Jetty and many more. Gems is another significant part of Ruby environment. Gem is a self-contained format for distributed ruby programs and libraries, which is managed by RubyGems package manager. For instance, current project uses the gem named "sqlite3" and can be replaced by the gem "pg", which is responsible for using the PostgreSQL database.

Ruby on Rails also uses the Model-View-Controller [22] design pattern to organize application programming. View part in Ruby on Rails is represented by Action View templates which are written using embedded Ruby in tags mingled with HTML. Controller part is represented by the Action Controller which is responsible for making sense of the request and producing the appropriate output. And finally, the Model part is represented by the Active Record design pattern. Active record is an architectural pattern that was named by Martin Fowler in his book Patterns of Enterprise Application Architecture. Active record object is an object that wraps a row in a database table or view, encapsulates the database access, and adds

domain logic on that data. An object carries both data and behavior. Much of this data is persistent and needs to be stored in a database. Active Record uses the most obvious approach, putting data access logic in the domain object. In this project this pattern is used in the server-side Ruby on Rails application as a part of the Object Relational Mapping system.

3.2.2 RESTful web services

Representational State Transfer (REST) is a coordinated set of architectural constraints that attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations. REST enables the caching and reuse of interactions, dynamic substitutability of components, and processing of actions by intermediaries, in order to meet the needs of an Internet-scale distributed hypermedia system [23] [24].

3.2.2.1 Client-Server

The first constraint is a separation of the user interface from the data storage concerns, which is improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components. This project holds this constraint by design.

3.2.2.2 Stateless

The next constraint is that a communication must be stateless in nature, such that each request from client to server must contains all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client. This constraint is also valid for this project. In the first phase of the development there is no user authentication, so each request is fully stateless. But in the future, user access also will be implemented in a stateless mode using the basic authentication or using authentication tokens.

3.2.2.3 Cache

The cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests. The advantage of adding cache constraints is that they have the potential to partially or completely eliminate some interactions, improving efficiency, scalability, and user-perceived performance by reducing the average latency of a series of interactions.

In this project cache is implemented using the request header "if-modified-since". The client application can add this header with a date of the last update into the request and get data, that changed since that date. In addition the Ruby on Rails on the server side automatically adds the field "updated_at" in each data entity and that is also ease a cache support.

3.2.2.4 Uniform interface

The central feature that distinguishes the REST architectural style from other network-based styles is its emphasis on a uniform interface between components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

3.2.2.5 Layered system

Layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot "see" beyond the immediate layer with which they are interacting. By restricting knowledge of the system to a single layer, we place a bound on the overall system complexity and promote substrate independence. In this project the client application interacts only with the application server which can be easily moved behind a load balancer. The database can be also detached on the separate server, but this will have no influence on the client applications.

3.2.3 Http Live Streaming

HTTP Live Streaming lets send audio and video over HTTP from an ordinary web server for playback on iOS-based devices - including iPhone, iPad, iPod touch, and Apple TV - and on desktop computers (Mac OS X). Also HTTP Live Streaming supported on other different platforms including Android and Windows Phone devices. HTTP Live Streaming supports both live broadcasts and prerecorded content (video on demand). HTTP Live Streaming supports multiple alternate streams at different bit rates, and the client software can switch streams intelligently as network bandwidth changes. HTTP Live Streaming also provides for media encryption and user authentication over HTTPS, allowing publishers to protect their work [4].

The HTTP Live Streaming supported not only by Apple devices. Android devices natively supports the HTTP Live Streaming from the Ice Cream Sandwich release (4.0) which is 90% of all devices [5]. Windows Phone devices can play HLS streams using 3rd party video players [6]. Among browsers only Safari natively supports HLS. Other browsers can play HLS throught the various 3rd party web players such as JWPlayer, which is currently used in the Czech American TV.

Main idea of HLS is a dividing video file on a series of small files, typically about 10 seconds duration, called media segment files. An index file, or playlist, gives the clients the URLs of the media segment files. Videos in the Czech-American TV are encoded using the H.264 video compression and use the AAC codec for audio. This encoders were chosen because of its support on the Apple devices and only that encoders are supported in the HTTP Live Streaming. So in this project there is no need to decode video. The simple script was created for segmentation video files. It uses the ffmpeg utility, that provides a set of portable, functional and high-performance libraries for dealing with multimedia formats of all sorts [26].

The HTTP Live Streaming has several advantages over progressive download.

- The first one is that HLS saves the video publisher and the viewer’s data plan money. The HTTP Live Streaming delivers only a few segments of the video which can leads to decreasing outgoing traffic and bandwidth load in the Czech American TV.

- Another advantage is that HTTP Live Streaming can switch between streams of different bitrates in response to changing connection speeds. Which means that on the cellular connection the mobile application can switch to the stream with low bitrate. That allows smooth playing of the video and also decreasing outgoing traffic from the server.

- The HTTP Live Streaming specification has provisions to ensure security of the stream. That will be very useful in the future for the Czech American TV membership program. Each media segment can be encrypted so the whole video stream is protected.

Chapter 4

Testing

Each product should be properly tested before the public release. For this projects two types of testing were chosen: usability testing with small group of people and more extensive beta testing with a lot of people.

4.1 Usability testing

Usability testing is used for detect any problems in the interaction between the user and the application. It allows developer to see weaknesses of the user interface on the early stages of the development. The usability testing for this project was made when the first design was implemented. Five people took part in it with different mobile using experience: 3 of them are Apple device users, 2 of them - Android device users.

There are several problems were discovered during testing and the main one is the drawer navigation. One of the iOS users had a problem with locating menu and at the beginning he thought that there is nothing in the application but the list of the videos on the main screen. Also, it took quite a significant amount of time for users to find a required section and often it leads to starting to scroll menu in an attempt to find it there.

Two testing users are developers and they both recommend to get rid of the drawer navigation. Also, in June were the Worldwide Developer Conference organized each year by Apple. In the session "Designing Intuitive User Experiences" User Ex-

perience evangelist Mike Stern talks about drawer navigation and its weaknesses. The first one is that menu is not visible on the screen at all. People that use applications with this pattern don't switch to different sections very frequently. And the main reason is that they can't see the options, or maybe they saw it at one point in time, but they have since forgotten. The next weakness is the speed and convenience in the section switching. Each time user should tap on the button in the corner, wait for the menu to open, find required section, tap on it. Also, menu button is located in the top left corner and on the mobile phones it is very hard to reach it using hand that handle the phone. Especially now, when Apple introduce the new iPhone with big screen size.

The most appropriate replacement for the drawer navigation is the tab bar. First of all, user always see what section is selected right now together with all other options to select. It is immediately clear, that the application has another features that can be selected. Which, is by the way, solves another problem of the drawer navigation - control reachability. Tab bars are located at the bottom of the screen so it can be easily reached. Switching between sections is fast and user has an immediate switch feedback from the application.

Another problem that was discovered during the testing is connected to the fact that Czech American TV service do not provide any additional information to the kitchen and classes videos. So in the detail view there are a lot of free space and users sometimes thought that the application didn't received the data yet.

These discovered problems lead to the big redesign of the application. Two test users from the first usability testing take part in the second testing with the new design and confirm that the application became much more simple and understandable.

4.2 Beta testing

The beta testing is the process of releasing the beta version of the product to the limited audience outside the programming team. In this project beta testing is combined with the term minimum viable product. The first release version of the project contains only core features that allow the product to be deployed. These core features here are downloading the list of the videos of all categories and their playback.

For collecting the information about the application installations Crashlytics ser-

vice was chosen. Crashlytics is powerful and lightweight crash reporting solution that also contains tools for the application analytics. It shows different performance metrics such as daily active users, average session length, crash-free users and more.

The beta version of the application was uploaded to the App Store and after 10 days was available for public. Basically, this step also includes some kind of testing, because all applications should pass a review by the Apple review team. First of all, reviewers control general stability and functionality: possible crashes, broken links, placeholder content and more. Besides review includes basic check of the application description page such as inaccurate description or incomplete information. Also, it verifies if the application follows Apple design guidelines, has clean, refined user-friendly interface.

The first beta testers were volunteers from the Czech American TV. About 10 people participate in the first phase of the testing and during first two weeks no crashes or some other serious problems have been detected. Which means that the core functionality such as video playback works fine.

Chapter 5

Conclusion

5.1 Current progress

The mobile iOS application project for the Czech-American TV is still in a progress and will be also develops after the presentation of this thesis.

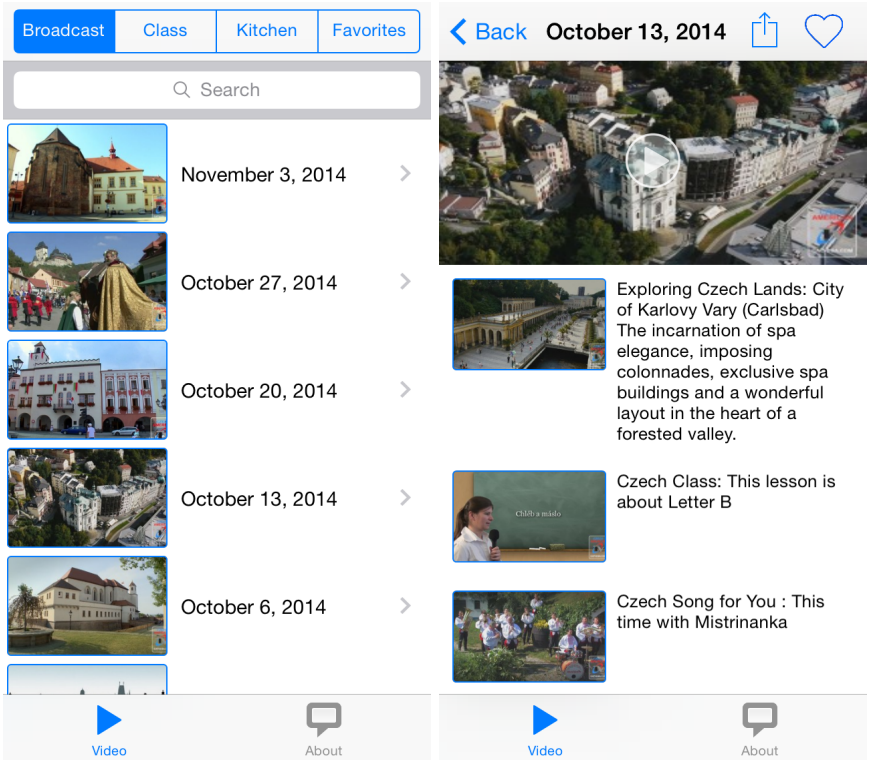

Figure 5.1: The iOS application: the tab bar with the segmented control (on the left) and the detail view (on the right)

The first public release of the iOS application is implemented according to redesigned application wireframes and shipped to the App Store. It includes only core video playback functionality, but all fundamental components of the application are almost ready for use and soon will be added to the application (Figure 5.2):

- The navigation in the application is realized with the tab bar pattern (Figure 5.1)
- The application use the Core Data stack and the local SQLite database as an application cache
- The application communicates with the server REST API through the HTTP protocol
- The application can play video streams from the server using the Http Live Streaming technology
- Points of interest can be displayed as a list or directly on the map
- A user can use the embedded Apple application Maps for turn-by-turn directions to the selected point of interest
- Quizzes and interaction with them are implemented

The server side application was implemented on the Ruby on Rails stack technology and now represented by the one web application:

- The server web application runs on the CTU's Openstack server and on the Heroku - cloud application platform
- The server uses the SQLite database as a data storage
- The application has the REST API for communicating with clients through the HTTP protocol
- The application API has a versioning support through the HTTP request's header "Accept". By default, the last version is available
- The application supports the HTTP Live Streaming including the client's authentication access permissions
- The content management system is implemented using the ActiveAdmin administration framework [25]

5.2 Future plans

The current project became quite large and needed a lot of implementation details for both sides - the client iOS application side and the server side. Although all project's fundamental parts were implemented, there are still a lot of work to do. First of all, the application should obtain all required functionality. It will be reached during the iterative process that includes following steps:

- Add an iPad support including platform-dependent distinctions
- Release in the AppStore version with integrated support of the points of interest
- Release in the AppStore version with integrated quiz support
- Implement radio playback support

Each step will go together with another public App Store release.

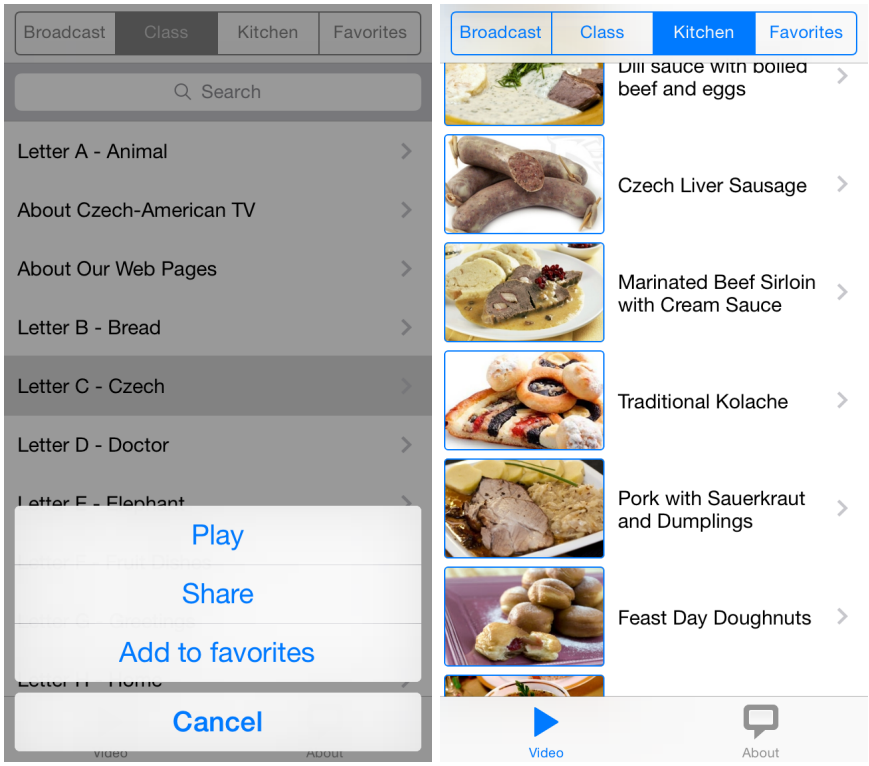

Figure 5.2: The iOS application: actions (on the left) and the kitchen view (on the right)

As regards the server side application, it is necessary to separate the database and the content server from the application server. Currently, in the Czech-American TV there is no authentication but they working on the new partnership system, where a user, that made a donation will get a full access to the content. As for the content management system, it needs to be customized for the current project, because now it is quite general and not so user-friendly and easy to use.

5.3 Final words

The mobile iOS application and the server application were implemented as a result of this project. In spite of the incompletenesses in both sides, the application's core functionality is complete. Two testing phases were accomplished during which some serious problems were discovered and fixed. The big redesign of the application was realized and led to much simpler and user-oriented design. And finally, as a big step from the development environment to the production - there was a public release in the App Store. Approve from the Apple reviewers to public the application can be considered as some kind of quality assurance.

In my opinion, current project will help the Czech-American TV in a popularization of the Czech culture not only in the USA but also all over the world. And due to the popularity of the Apple's online application store Appstore there is a high probability of the increasing Czech-American TV users and members and, as a result, more resources and opportunities for the project's development and evolution.

References

[1] J. Conway and A. Hillegass. *iOS Programming: The Big Nerd Ranch Guide*. Big Nerd Ranch Guides; 4 edition (February 21, 2014).

[2] *Apple devices statistics.* URL: https://developer.apple.com/support/appstore/

[3] *Auto Layout Guide* URL: https://developer.apple.com/library/ios/documentation/userexperience/conceptual/AutolayoutPG/Introduction/Introduction.html

[4] *HTTP Live Streaming* URL: https://developer.apple.com/streaming/

[5] *Android Platform Versions* URL: http://developer.android.com/about/dashboards/index.html

[6] *HTTP Live Streaming Client SDK for Windows 8 and Windows Phone 8* URL: http://www.3ivx.com/technology/windows/metro/http_live_streaming.html

[7] *In-App Purchase for Developers* URL: https://developer.apple.com/in-app-purchase/

[8] *Map Kit Framework* URL: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/MapKit/MapKit.html

[9] Matt Thompson *UIActivityViewController*. URL: http://nshipster.com/uiactivityviewcontroller/

[10] *Ruby on Rails* URL: http://rubyonrails.org/

[11] *SQLite* URL: https://sqlite.org/

[12] *JetBrains AppCode* URL: https://www.jetbrains.com/objc/

[13] *Apple Push Notification Service* URL: https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html

[14] *UML, Technopedia* URL: http://www.techopedia.com/definition/3243/unified-modeling-language-uml

[15] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[16] *Apple Model-View-Controller Design Pattern* URL: https://developer.apple.com/library/ios/documentation/general/conceptual/devpedia-cocoacore/MVC.html

[17] *A*pple Singleton Design Pattern URL: https://developer.apple.com/library/mac/documentation/general/conceptual/devpedia-cocoacore/Singleton.html

[18] *A*ndroid Navigation Drawer Design Pattern URL: https://developer.android.com/design/patterns/navigation-drawer.html

[19] *G*itHub MMDrawerController project URL: https://github.com/mutualmobile/MMDrawerController

[20] *A*pple Delegate Design Pattern URL: https://developer.apple.com/library/mac/documentation/general/conceptual/devpedia-cocoacore/Delegation.html

[21] *A*pple Core Data Programming Guide URL: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/cdProgrammingGuide.html

[22] *R*uby on Rails MVC Principles URL: http://guides.rubyonrails.org/getting_started.html

[23] Roy Fielding (2000). *Architectural Styles and the Design of Network-based Software Architectures*. URL:http://www.ics.uci.edu/~{}fielding/pubs/dissertation/top.htm

[24] Roy Fielding, Richard Taylor *P*rincipled Design of the Modern Web Architecture URL: http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf

[25] *A*ctiveAdmin administration framework URL: http://activeadmin.info/

[26] *F*Fmpeg URL: https://www.ffmpeg.org/

[27] *A*ppropriate Uses For SQLite URL: http://www.sqlite.org/whentouse.html

Appendixs A

Video adaptation manual

The Http Live Streaming technology has following requirements [4] for the video files:

- Video should be encoded using the H.264 compression
- Audio tracks should be encoded either HE-AAC, AAC-LC or MP3

Currently, all video files in the Czech American TV answer these requirements. Next, all video files should be divided into segments. This can be reached with help of the FFmpeg utility [26]. It is free and cross-platform, but in case of compiling from the source code it should be configured with flags "–enable-gpl –enable-libx264". For instance, we have a video file with name "video.mp4" and we want to create a HLS playlist with name "hls.m3u8". The following command split video file into several segment files (hls0.ts, hls1.ts, etc.) and create the playlist file (hls.m3u8):

```
ffmpeg -i video.mp4 -strict -2 hls.m3u8
```

The AAC encoder is still experimental so the command should contains flag "-strict -2".

Now the video is ready for streaming. The HTTP Live Streaming can be served from any web server so no special configuration is necessary except the MIME types of the streaming files that will be served. For the playlists *.m3u8 files MIME Type should be "application/x-mpegURL" or "vnd.apple.mpegURL". For the segment *.ts files it should be "video/MP2T". Because m3u8 files are simple text files they can be compressed with the gzip compression - the mobile application can automatically unzip them.

For the mobile application is enough to send a link to the m3u8 playlist file of the video instead of the link to the original mp4 file. But due to advantages of the HLS it can be also integrated into the current website.

```
<video
    src="http://path/hls.m3u8"
    height="300" width="400"
>
<!— fallback block —>
</video>
```

This code snippet add the HTML5 tag "video" with the link to the m3u8 playlist. In case that browser don’t support the video tag or don’t support the HTTP Live Streaming fallback block can be included between <video> and </video> tags.

Appendixs B

REST Service Interface

```
Resource: Videos

GET /videos/  // return all videos
Response format: [
{
        "video_id" : 1,
        "category_id" : 1,
        "url" : "http://URL/playlist.m3u",
        "name" : "Video name",
        "description" : "Video description",
        "date" : 1394478580
},
...
]


GET /videos/categories /// return all video categories
Response format: [
{
        "category_id" : 1,
        "name" : "Category name",
        "kind" : "video",
        "level" : 0
},
...
]

```

```
GET /videos/categories/1/videos /// all videos in category
Response format: same as "GET /videos"


GET /videos/categories/:category_id /// return category by id
Response format: one category

GET /video/:video_id /// return video by id
Response format: {
        "video_id" : 1,
        "category_id" : 1,
        "url" : "http://URL/playlist.m3u",
        "name" : "Video name",
        "description" : "Video description",
        "date" : 1394478580
}


GET /video?search="search_term"  /// search in all videos
Response format: same as "GET /video"


Resource: Radio

GET /radios /// return all radio streams
Response format: [
{
        "radio_id" : 1,
        "url" : "http://URL/radio.m3u",
        "name" : "Radio stream name",
        "description" : "Radio stream description",
        "date" : 1394478580
},
...
]

GET /radios/:radio_id /// return radio stream by id
Response format: {
        "radio_id" : 1,
        "url" : "http://URL/radio.m3u",
        "name" : "Radio stream name",
        "description" : "Radio stream description",
        "date" : 1394478580
```

```
}


Resource: POI

GET /pois /// return all pois
Response format: [
{
        "poi_id" : 1,
        "category_id" : 1,
        "latitude" : 50.083333,
        "longitude" : 14.416667,
        "name" : "POI name",
        "description" : "POI description"
        "address" : "T r i t? 365/15 118 00 Praha 1-Mal  Strana Czech
           Republic"
},
...
]

GET /pois/categories /// return all POI categories
Response format: [
{
        "category_id" : 1,
        "name" : "Category name"
},
...
]

GET /pois/categories/:category_id /// return category by id
Response format: same as "GET /poi"

GET /pois/:poi_id /// return POI by id
Response format: {
        "poi_id" : 1,
        "category_id" : 1,
        "latitude" : 50.083333,
        "longitude" : 14.416667,
        "name" : "POI name",
        "description" : "POI description"
        "address" : "Address"
}

```

```
114
115 Resource: Quizzes
116
117 GET /quizzes /// return all quizzes
118 Response format: [
119 {
120         "qiuz_id" : 1,
121         "name" : "Quiz name",
122         "date" : 1394478580
123 },
124 ...
125 ]
126
127
128 GET /quizzes/:quiz_id /// return quiz by id
129 Response format: [
130 {
131         "question" : "Questions text",
132         "answers" : [
133                 {
134                         "answer_id" : 1,
135                         "text" : "answer text",
136                         "isCorrect" : 0
137                 },
138                 {
139                         "answer_id" : 2,
140                         "text" : "answer text",
141                         "isCorrect" : 1
142                 },
143                 ...
144         ]
145 },
146 ...
147 ]
```

Appendixs C

Application wireframes for tablets

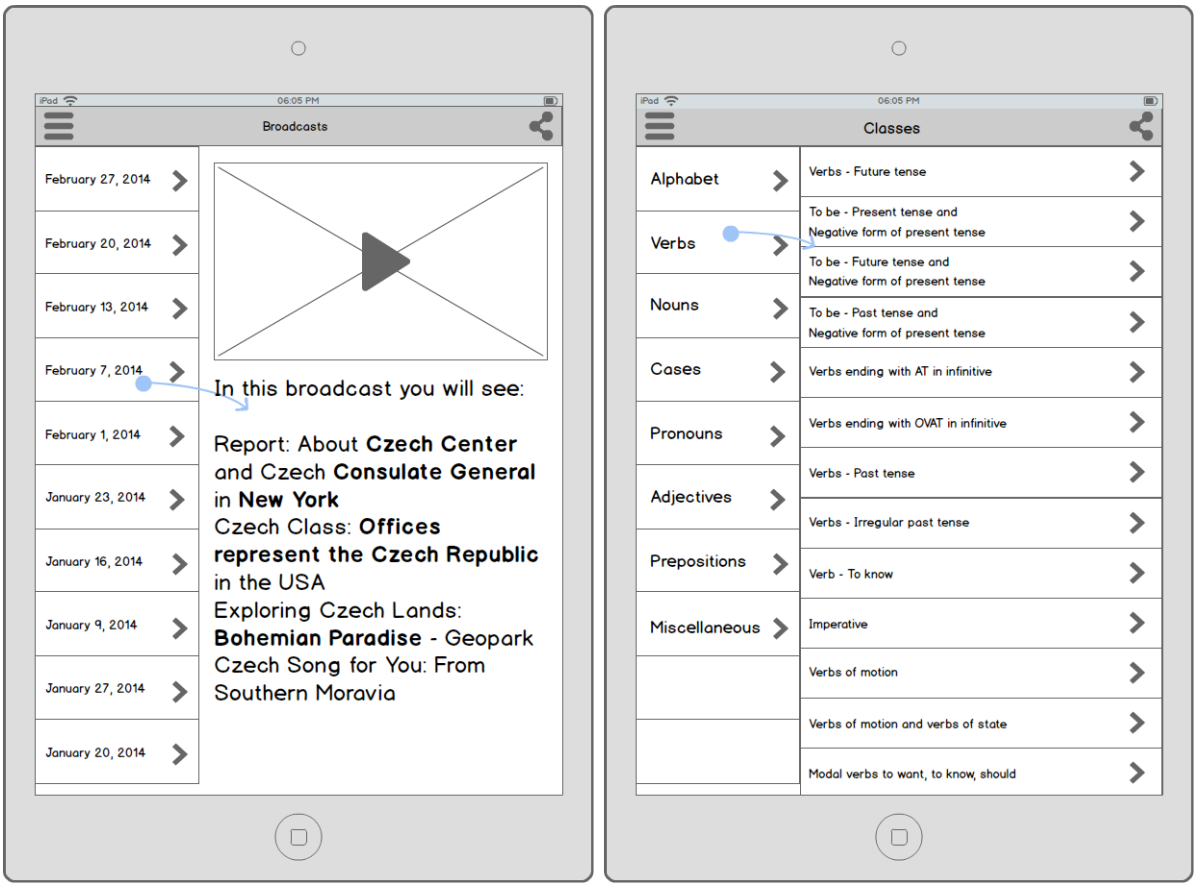

Figure C.1: Tablet wireframes: video controllers

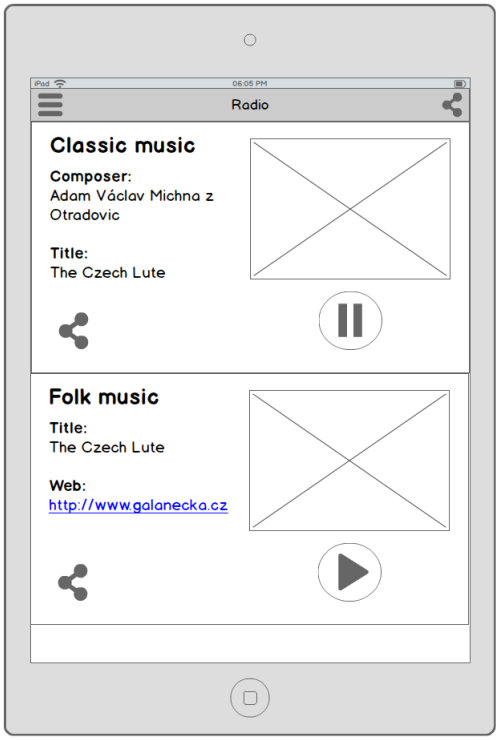

Figure C.2: Tablet wireframes: radio controller

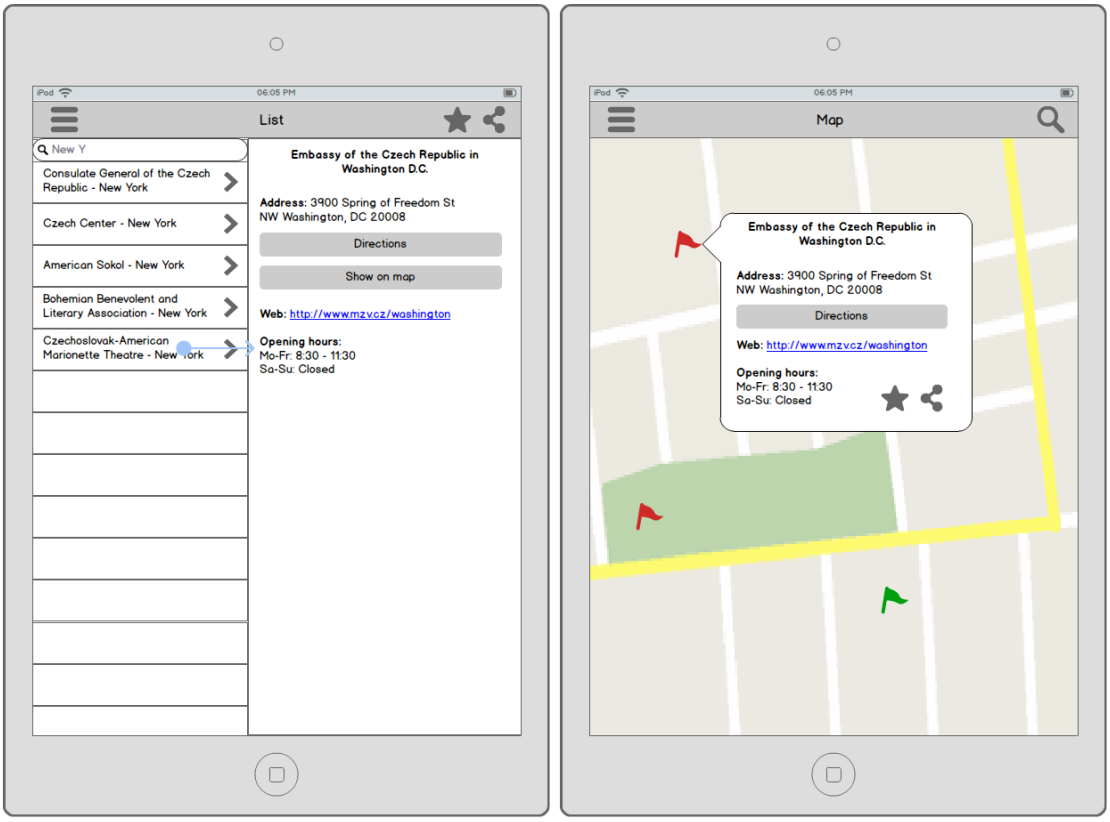

Figure C.3: Tablet wireframes: POI controllers