BABES-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

COMPUTER SCIENCE, ENGLISH

DIPLOMA THESIS

Virtual Boards and Server-less Streaming

Supervisor:

Bogdan Pop

Author:

Tudor Filip Stupariu

2018

# Abstract

This paper presents a concept platform that allows its users to create different rooms, public or private, and create content via graphics and text. Next, the content is transmitted in real time to other users via a video stream capturing the graphics being drawn. The platform also allows for users to comment in real time using a chat system. The platform consists of a web application based on an API created in Ruby on Rails and a front-end client based on VueJS.

This paper is structured as follows. Chapter 1 provides an introduction, the motive behind creating this application and why this solution is needed. Chapter 2 presents some current possible solutions to the problem described in Chapter 1, and analyses their issues and faults. Chapter 3 goes through the main solutions introduced by the application and why they are superior to current solutions. Chapter 4 presents the application in details, both from a theoretical aspect and from a practical aspect, also providing samples of code throughout. Chapter 5 brings a conclusion and a short summary of the solution presented, while Chapter 6 contains the bibliography and inspiration used for the paper.

# CONTENTS

1. Introduction

* 1.1 Problem definition
* 1.2 Web based graphics and server-less streaming  
    
  1.3 General description and personal contribution

1. Current Theoretical Approaches

* 2.1 Graphics and virtual boards
* 2.2 Streaming approaches  
    
  2.3 Problems with current solutions  
    
  2.4 Solution to the problems

1. Benefits of proposed solution
2. Application details

* 4.1 Functionalities
* 4.2 Design  
    
   4.2.1 Backend Server  
    
   4.2.2 Database  
    
   4.2.3 Streaming Solution  
    
   4.2.4 Web Graphics  
    
   4.2.5 Serverless Computing  
    
   4.2.6 UI and Front-end  
    
  4.3 Implementation  
    
   4.1.1 Backend Server  
    
   4.3.2 Database  
    
   4.3.3 Streaming Solution  
    
   4.3.4 Web Graphics  
    
   4.3.5 Serverless Computing  
    
   4.3.6 UI and Front-end  
    
  4.4 Validation and Testing  
    
  4.5 Personal Contribution

1. Conclusion
2. Bibliography

# 1. INTRODUCTION

This chapter is aimed at providing a short inside of the problem this platform is trying to solve and how/why it's doing things the way it currently is implemented.

## 1.1 Problem Definition

Collaboration and remote teaching tools have existed for a ling time, but they have always been platform dependent or hard to use. A lot of what we're doing on a day to day basis is on the internet, so it was just a matter of time until collaboration tools would go to a web only approach. The problem of drawing graphics on the web has always existed and solutions have been around like Adobe Flash, but they were not always reliable and lately they have been lacking support. Since the introduction of HTML5, a new element was added called "canvas" and along with it an API you can interact with using JavaScript.

Another problem has been streaming while keeping costs low. Big platforms like Twitch have been doing it using intermidiate servers to handle the stream data and distribution, but for small groups of people this is a really costly solution, since the processing power required by the servers in incredibly high.

Combining the two, remote learning and teaching has always been a bit of a challenge, and this platform aims to make it easy to use and integrate seamlessly in the day-to-day life of its users, while also providing enough tools for someone to manage to be productive using the app.

## 1.2 Web Based Graphics and Serverless Streaming

Drawing graphics on the web isn't really a challenge anymore, the problem comes when you have to add user interaction. Just displaying shapes has always been decently easy, but making a user dynamically draw shapes is not the easiest of tasks. Also, former solutions involved plugins which some users may not have installed or it may confuse a user about why the platform does not seem to work. This solution provides a general solution using a standard provided by HTML itself.

Serverless streaming, or peer-to-peer streaming is also a new idea on the web. P2P file transfers have existed for a long time, but just recently there has been introduced a standard for managing such connections that also support data, video and audio streams in the form of WebRTC P2P streaming. This is a standard/API that can easily be accessed by JavaScript on the web, providing a general solution for any user on any platform, without requiring extra plun-ins or intermediate servers ti handle tha data, which might introduce more latency in the stream feed.

## 1.3 General Description and Personal Contribution

This application provides a simple to use web client accessible by any user through a web browser. It provides authentication in order to keep the platform secure, and allows the user to create both public and password protected private groups. Once in a group, the initiator can start the stream, while the other attendees can start watching a video feed. The initiator has a canvas on which he can freely draw graphics, text, images and shaped and that information is transmitted live through the video feed. It also provides a form of live interaction through a chat, so that the attendees can freely ask questions or leave comments.

I have personally developed the API for the web client and implemented the authentication, database structure and back-end logic. Also, I created the web client and layout of the interface. The logic behing signalling peers and live chat has been implemented from scratch, as well as all the tools for drawing on the canvas combined with the user interaction with these tools.

# 2. CURRENT THEORETICAL APPROACHES

This chapter describes the most used current approaches to the problem. It also goes a little bit into each approach and it's downsides for our goal of achieving a low cost and easy to use streaming and drawing learning platform.

## 2.1 Graphics and virtual boards

Currently, there are multiple other approaches for drawing on a blank canvas on the web, without the need for native clients. Some of these include SVG graphics and Flash drawing.

SVG graphics work pretty well performance-wise. Also, since they are vector graphics, they scale really well. The problem is, we don't only want to display graphics, but we want a user to be able to create them. And here, the tooling with SVG gets a little trickier. The handling of shapes, text or other types of primitive graphics is harder to implement and doesn't make too much sense in an environment like SVG.

The other option would be Flash, but the problem is that it is an old technology that a started to lack support in the recent years. First, you do need a plugin for running flash applications, which makes it much less user friendly. Also, Adobe, the company offering the support for this technology, said they would drop support of it by the year 2020 [<https://support.google.com/chrome/answer/6258784?co=GENIE.Platform%3DDesktop&hl=en>].

## 2.2 Streaming Approaches

Streaming in general is an action that requires a lot of resources, both computational and network ones. Traditionally, streaming has been done through standalone servers user exactly for that purpose. Whlie they do produce decent scalability for a big number of concurrent users trying to watch a stream, they also introduce a couple problems, the main one being cost. Since it is a resource hungry activity, the server cost is really high. This way, mantainance and scalability can be quite costly. Also, since this application is made for small rooms of people, it will not directly benefit from the ability to handle a great amount of users watching a stream at the same time.

## 2.3 Problems with current solutions

Current solutions are either too old, cumbersome or expensive to implement and mantain. Server resources are expensive, and the more processing that can be done on the client, the cheaper the mantainance cost of the application will be. Having more servers and nodes that a stream has to pass through can also introduce more latency laong the line. Also, making the user interaction as straight-forward as possible is not so wasy using these technologies.

## 2.4 Solutions to the problems

Each problem has a solution, starting with the graphics. During the launch of HTML5, a new element and API was introduces called Canvas. It offers you a place that does not rely on a DOM for creating graphics. It also needs no other external tool or extension, the interaction and drawing being done solely reliant on Javascript. It's main advantages are ease of use and the effortless user interaction it can provide. Also, this API offers an easy way to capture a video stream of the actions and graphics being drawn.

For the streaming part, the solution for expensive server resources is eliminating servers all together. WebRtc P2P video streaming is a technology optimized for streaming data, including video, between multiple peers directly. This way, the system only needs a server for signalling clients when a connection offer is coming, but no actual video stream data passes through it. The connection for the video stream is made directly between the 2 clients' browsers, minimizing the server resources needed.