UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

TESIS PARA OBTENER EL GRADO DE:

LICENCIADO EN TECNOLOGÍA

TÍTULO:

DISEÑO Y DESARROLLO DE UNA APLICACIÓN PARA EL APRENDIZAJE COLABORATIVO EMPLEANDO TECNOLOGÍAS WEB.

PRESENTA

ERIC VALDEZ VALENZUELA

DIRECTOR DE TESIS

DR. FERNANDO GAMBOA RODRÍGUEZ

AGOSTO 2015

Juriquilla, Querétaro

A mi familia,
a mis profesores,
a mi tutor y sinodales,
a mis amigos,
a la primaria Christel House
y a mi universidad

Gracias.

ÍNDICE

INTRODUCCIÓN

En este trabajo de tesis se presenta el diseño y desarrollo de una aplicación web para el aprendizaje colaborativo, cuyo principal objetivo es ampliar y mejorar las posibilidades de uso de la aplicación “escritorio colaborativo” desarrollada en el proyecto “el Aula del Futuro”, del Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET, UNAM). En el Aula del Futuro se desarrollan tecnologías que buscan integrarse a espacios de aprendizaje flexibles, en respuesta al dinámico contexto tecnológico actual que se ha diversificado, trayendo nuevos retos y oportunidades.

Se espera que el desarrollo de esta aplicación, que se ubica dentro del área del Aprendizaje Colaborativo Asistido por Computadora o CSCL (*Computer-Supported Collaborative Learning*), sea la base para la creación de futuros espacios colaborativos interactivos empleando tecnologías web y que su uso se extienda a casos prácticos y sean implementadas en las aulas y otros espacios educativos.

El trabajo aquí desarrollado permitió generar una aplicación que habilita un espacio colaborativo donde los usuarios pueden generar y compartir información mientras realizan una actividad en común, teniendo como efecto entre los integrantes del equipo la adquisición de un conocimiento más amplio de un concepto y/o la mejora de las habilidades para trabajar en equipo.

La aplicación fue probada en una primaria con alumnos de cuarto y quinto grado mediante la participación en el “Proyecto Piloto de Inclusión Digital” que forma parte de la Estrategia Nacional Digital propuesta por Presidencia del Gobierno federal de México. En estas pruebas se encontró que durante el uso de la herramienta se promovió la toma de acuerdos, las discusiones y las negociaciones, elementos que fundamentan el aprendizaje colaborativo.

El contenido de este trabajo se organiza de la siguiente manera: en el capítulo 1 se presenta el problema a resolver, en los capítulos 2 y 3 se expone una introducción al aprendizaje colaborativo y las tecnologías web, en los capítulos 4 y 5 se describe la arquitectura diseñada para la aplicación, así como ejemplos de las variables y funciones fundamentales

del código, en el capítulo 6 se presentan las pruebas realizadas con el usuario final y los análisis a los resultados. Por último en el capítulo 7 se redactan las conclusiones obtenidas al finalizar este trabajo de tesis.

I.I Hipótesis

Es factible multiplicar y potenciar los beneficios de los espacios colaborativos a partir de un uso más extenso de las tecnologías web y las facilidades que estas proporcionan. Esta flexibilidad tendrá una repercusión positiva en la cantidad de escenarios educativos que es posible prever y por ende se propone desarrollar aplicaciones web que mejoren las habilidades de planeación y trabajo en equipo en alumnos mediante una actividad recreativa.

I.II Justificación

Se busca crear un espacio relacionado al aprendizaje colaborativo similar a los existentes en el “Aula del Futuro” mediante la implementación de tecnologías web de código abierto, ya que el uso de estas tecnologías permite crear aplicaciones multiplataforma, portátiles y de bajo costo al ser de distribución libre.

I.III Objetivo General

Diseñar y desarrollar una aplicación web con el propósito de ampliar y mejorar las posibilidades de uso del escritorio colaborativo desarrollado en el CCADET.

I.IV Objetivos Generales

- Diseñar la aplicación web a partir de las propuestas existentes en el Aula de Futuro.
- Analizar y definir las tecnologías web a emplear.
- Definir y desarrollar la arquitectura de software en el lado del cliente.
- Definir y desarrollar la arquitectura de software en el lado del servidor.
- Realizar pruebas para verificar el funcionamiento correcto de la aplicación.
- Realizar pruebas con el usuario final

CAPÍTULO 1

DESCRIPCIÓN DEL PROBLEMA

El grupo interdisciplinario de Espacios y Sistemas Interactivos para la Educación (ESIE), del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM, ha desarrollado desde el 2007 el proyecto “El Aula del Futuro” el cual busca la creación y el estudio de espacios de aprendizaje flexibles, colaborativos y enriquecidos con tecnología. El objetivo del grupo es diseñar, evaluar y estudiar propuestas que permitan entender los criterios involucrados en la integración de la tecnología al proceso educativo de manera exitosa.

Una de las propuestas que mejores resultados ha tenido dentro del Aula es el Escritorio Colaborativo Aumentado (ECA), el cual es un espacio de trabajo que permite a cuatro usuarios generar e intercambiar información digital proyectada y manipulada en un área en común, que podría ser un escritorio o una pared. Así, cada uno de los usuarios del ECA tiene acceso a un espacio privado de trabajo (su computadora) y un espacio público compartido (el proyectado sobre un espacio en común). Es en este último donde se realiza la toma de acuerdos, las discusiones y las negociaciones que fundamentan el aprendizaje colaborativo.

El ECA está compuesto por una computadora encargada de proyectar la parte correspondiente al “Escritorio” o el espacio compartido; cuatro computadoras personales (una para cada usuario), y un servidor encargado de sincronizar la información proveniente tanto de las computadoras de los usuarios, como de la computadora responsable de presentar el escritorio. El principio de uso es simple: un usuario puede compartir documentos desde su computadora (su espacio privado) hacia el escritorio (el espacio público), haciéndolos accesibles al resto de los participantes. De esta manera, todo lo que es proyectado sobre el escritorio es susceptible de ser manipulado y modificado por cualquiera de los cuatro usuarios, logrando con esto uno de los principios básicos de la propuesta, que es el control distribuido; esto es, los cuatro usuarios son dueños de lo proyectado en el espacio público. Esto es posible gracias a un sistema de comunicación en el que la computadora responsable de la proyección en la zona pública recibe mensajes del servidor, en los que le comunica las acciones que está realizando cada usuario en su computadora. En la figura 1.1 se muestra a un grupo de cuatro estudiantes haciendo uso del ECA.

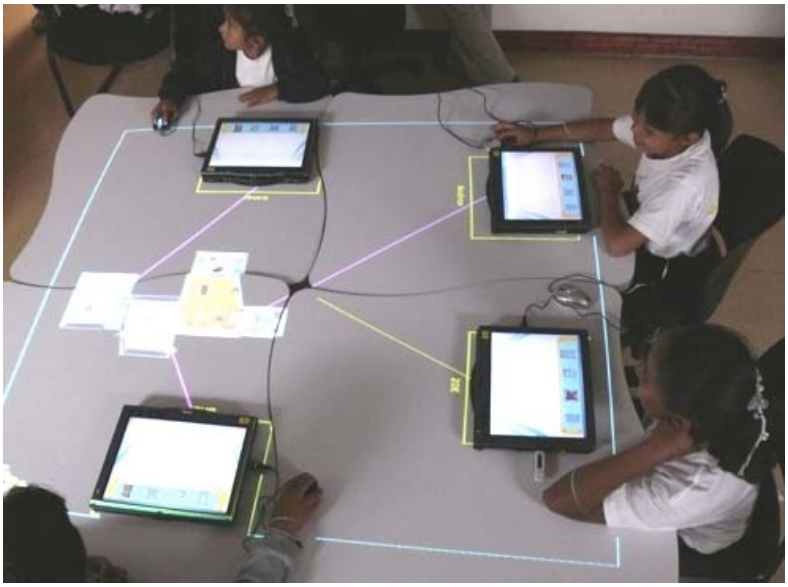


Figura 1.1 Alumnos usando la aplicación Escritorio Colaborativo Aumentado

Para el intercambio y sincronización de información entre las computadoras que intervienen en el ECA, se emplea un sistema de comunicación cliente-servidor basado en la tecnología Flash Server, y una serie de aplicaciones que únicamente funcionan en Windows. Es aquí donde se presenta el problema, la dependencia a un sistema operativo específico pone en desventaja a la aplicación en un mundo donde crece el número y uso de los dispositivos móviles, los cuales demandan aplicaciones multiplataforma.

En las últimas décadas hemos experimentado un vertiginoso avance en el desarrollo de las tecnologías de la información por lo que hoy en día existe un amplio espectro de tecnologías multiplataforma libres a emplear para eliminar esta limitante al ECA desarrollado en el aula del futuro. Por ello se propuso desarrollar una aplicación con herramientas que brinden la ventaja multiplataforma, como lo son la mayoría de las tecnologías web actuales. Además se busca crear una arquitectura reutilizable que pueda ser empleada en un futuro para el desarrollo de más aplicaciones relacionadas con el aprendizaje colaborativo y sirvan como espacios más flexibles que no dependan de una plataforma o sistema operativo en particular y que se puedan utilizar en cualquier dispositivo que cuente con un navegador web, ya sea una computadora de escritorio o portátil, dispositivos móviles o incluso en sistemas embebidos de bajo costo como un *Raspberry Pi.*

Como solución, se propone emplear las siguientes tecnologías: un servidor *node js* para sincronizar la información entre la zona pública y la zona privada; la librería *socket.io* para crear un canal de comunicación abierto entre un cliente y un servidor; *Express* como marco de trabajo para el desarrollo de la aplicación; *HTML5, CSS3* y *Javascript* para el interfaz y funcionalidad del lado del cliente. Todas estas propuestas son de código abierto y multiplataforma.

A continuación se presenta de manera breve el procedimiento y tecnologías a utilizar para el desarrollo de la aplicación web:

- Diseñar un prototipo de funcionalidad (*mockup*).
- Analizar y definir la arquitectura de software y las tecnologías a emplear en el lado del cliente (*front-end*) y en el lado del servidor (*back-end*).
- Programar la aplicación web bajo el lenguaje de programación *JavaScript* principalmente, empleando el editor de textos *Sublime Text.*
- Realizar pruebas de funcionalidad a la aplicación web.
- Analizar los resultados, errores y mejoras realizadas al prototipo.

CAPÍTULO 2

APRENDIZAJE COLABORATIVO

La definición más general de aprendizaje colaborativo se describe como la situación en la cual dos o más personas aprenden o intentan aprender algo juntos [1]. Cada uno de estos elementos (el número de personas, el aprendizaje y la colaboración) puede ser interpretado de diferentes maneras:

• **Número de personas:** puede ser interpretado como un grupo pequeño de dos a cinco personas, una clase de alumnos de 20 a 30 personas, una comunidad de dos mil personas o como una sociedad de varios miles o millones de personas, y así como todos sus niveles intermedios.

• **El aprendizaje:** Puede tener diferentes interpretaciones, como por ejemplo: seguir un curso, estudiar el material de una clase, desempeñar actividades de aprendizaje mediante la resolución de problemas y aprender mediante un trabajo en equipo o una práctica.

• **La colaboración**: Puede referirse a diferentes formas de interacción, cara a cara, mediante una computadora, de manera sincrónica o asincrónica, mediante un esfuerzo en conjunto o dividiendo las actividades de una manera sistemática.

Estos tres elementos definen y delimitan el universo de lo que se considera aprendizaje colaborativo. En esta tesis definiremos un caso concreto de aprendizaje colaborativo: el que se presenta en un salón de clases donde grupos de dos a seis personas generan e intercambian información para lograr un objetivo en común.

2.1 Aprender colaborando

En la literatura científica se considera el aprendizaje colaborativo como una actividad en la que un grupo de personas se ven involucradas en la resolución de un problema y en la cual ocurre un aprendizaje como efecto secundario y este puede ser medido mediante la adquisición de nuevo conocimiento o por la mejora en las habilidades para resolver problemas en equipo [1]

Es importante aclarar que el aprendizaje colaborativo no se puede considerar un mecanismo aislado de aprendizaje. Si se habla del aprendizaje colaborativo, entonces también se debe de hablar del aprendizaje individual. El sistema cognitivo individual no aprende por el hecho de ser individual, aprende por la realización de ciertas actividades

(leer, construir, predecir, deducir, etc) que desencadenan mecanismos de aprendizaje (inducción, deducción, compilación, etc). Similarmente, los integrantes de un equipo no aprenden por el hecho de ser dos o más personas, si no por que realizan ciertas actividades (explicar, discutir, comprender otros enfoques, etc) que provocan a su vez la presencia de ciertos mecanismos de aprendizaje (obtención de conocimiento, socialización, reducción de carga cognitiva, etc).

Por otro lado, el aprendizaje colaborativo tampoco se puede considerar un método debido a la baja predictibilidad de que ocurran ciertas interacciones que traen consigo mecanismos de aprendizaje. Es decir, en la práctica el aprendizaje colaborativo toma forma de un contrato social donde se especifican reglas que se deben de cumplir, estas normalmente son: la forma en que interactúan los integrantes (p. ej. “tienen que trabajar juntos”), el lugar donde se trabaja (p. ej. “los integrantes de los equipos deben trabajar en la misma mesa”) y las restricciones (p. ej. “cada integrante debe de contribuir a la conclusión”). En este contrato social se pueden dar interacciones que disparan mecanismos cognitivos de aprendizaje, mas no hay garantía de que estas ocurran.

Entonces, considerando que el aprendizaje colaborativo no es un mecanismo ni un método, un mejor enfoque para su definición sería observarlo como una situación en la se espera que ocurran ciertas formas de interacción (explicación, discusión, auto-regulación) en un grupo de personas, y que estas interacciones a su vez promuevan ciertos mecanismos de aprendizaje (inducción, deducción, compilación).

Se han establecido componentes esenciales para que un entorno se considere colaborativo [2]:

- **Interdependencia positiva**: Lo que afecta a un integrante del grupo de manera positiva o negativa afecta también a los demás integrantes. Para lograr el objetivo del grupo, cada integrante debe alcanzar los propios. Se comparten recursos, proporcionando apoyo mutuo.
- **Interacción cara a cara:** La importancia de este componente radica en que algunas actividades cognitivas y dinámicas interpersonales ocurren únicamente con la interacción directa de los integrantes del grupo, además permite obtener realimentación entre los miembros del grupo.

- **Valoración y responsabilidad personal:** La evaluación del avance personal es indispensable para conocer el desempeño individual de los integrantes del grupo y de esta forma tener indicios del valor del trabajo grupal.
- **Habilidades interpersonales y de equipo:** A los miembros del grupo se les debe de enseñar las habilidad sociales y motivarlos a emplearlas, ya que son necesarias para alcanzar una colaboración enriquecida.
- **Evaluación grupal:** Es conveniente realizar un análisis del trabajo que se está realizando en el grupo, identificando las diferentes acciones, con el objetivo de tomar decisiones relacionadas con el proceso de trabajo que se está llevando.

2.2 Teoría del aprendizaje colaborativo

La teoría que sustenta el aprendizaje colaborativo establece que se deben de considerar cuatro criterios en un escenario didáctico para aumentar la probabilidad de que exista un aprendizaje colaborativo [1]. Estos criterios son:

- El conjunto de elementos que crean una **situación** apropiada para el aprendizaje colaborativo (similitud entre los integrantes del grupo, grado de repartición de tareas).
- Las **interacciones** que provocan mecanismos de aprendizaje (negociación, sincronización, interactividad).
- Los **procesos** de aprendizaje (modelado mutuo de un problema, ampliación de conceptos).
- Los **efectos** del aprendizaje colaborativo (mejora de habilidades de trabajo en equipo, comprensión más amplia de un concepto).

Lo anterior se puede visualizar en la secuencia mostrada en la figura 2.2.1. Se puede resumir de la siguiente manera: una situación adecuada para el aprendizaje colaborativo puede generar ciertos patrones de interacción que a su vez provocan mecanismos cognitivos o procesos de aprendizaje, los cuales tienen efectos cognitivos.

Figura 2.2.1 Criterios involucrados en el aprendizaje colaborativo

En las siguientes secciones se describen con más detalle cada uno de estos criterios.

2.3 Situaciones colaborativas

Para que una situación o escenario educativo promueva el aprendizaje colaborativo deben de cumplir con las siguientes condiciones: las personas en el grupo deben de tener niveles de conocimiento similares, realizar las mismas tareas, tener una meta en común y trabajar juntos [1]. Examinando estos cuatro criterios, se encuentra que debe de existir ciertas similitudes entre los integrantes del grupo y las actividades que realizan, a estas similitudes las llamaremos simetrías. Existen diferentes formas de simetría:

- **Simetría de acción**. Es el grado en que se permite que las personas del grupo realicen las mismas acciones.
- **Simetría de conocimiento**. Es el grado en que las personas del grupo poseen los mismos conocimientos. Dos integrantes pueden tener el mismo grado de experiencia pero diferentes puntos de vista de un concepto.
- **Simetría de estatus**. Hace referencia al grado de similitud entre los estatus de las personas del grupo.

Las simetrías en una situación colaborativa afectan directamente la forma en que van a interactuar los integrantes, por ejemplo una persona puede creer que su compañero es más experto en un tema por lo que este adopta una posición débil en su argumentación. Sin embargo, en algunos escenarios es deseable que exista una pequeña asimetría en el conocimiento entre los pares ya que esto lleva a interacciones que derivan en discusiones.
Una vez que se han previsto los elementos que deben de haber en una situación colaborativa, se espera que con estos aumenten las probabilidades de que ocurran ciertas interacciones.

2.4 Interacciones colaborativas

Existen principalmente tres elementos que influyen en las interacciones colaborativas: interactividad, sincronización y negociación [1]:

- **Interactividad**: Indica el grado de interacción entre los pares, este no está definido por la frecuencia de sus interacciones, si no más bien por cómo estas interacciones influyen en los procesos cognitivos de los integrantes del grupo.
- **Sincronización**: Realizar una actividad en equipo implica comunicación sincronizada. Este elemento se puede considerar como una regla de comunicación en la que un integrante del grupo emite un mensaje y este espera a que su par lo reciba y lo procese tan rápido como le sea llegado. Dicho de otra manera, se busca un razonamiento sincronizado por parte de los integrantes.
- **Negociación**: Se espera que en una situación colaborativa los integrantes del equipo puedan argumentar su punto de vista, justificarlo, negociar e intentar convencer al otro integrante.

Estas interacciones pueden disparar procesos que derivan en un aprendizaje.

2.5 Procesos colaborativos que guían al aprendizaje

.

Ahora pasaremos a definir cuales son los mecanismos específicos del aprendizaje colaborativo. Es importante aclarar que a nivel fundamental los mecanismos de aprendizaje involucrados en el aprendizaje colaborativo son similares a los que se presentan en el aprendizaje individual. A continuación se listan mecanismos de aprendizaje que se pueden presentar en interacciones colaborativas:

- **Inducción**: La inducción crea representaciones más abstractas de un problema o un concepto, esto se debe a que mediante la interacción entre miembros de un grupo surge una nueva definición que integra las características o definiciones de cada persona.
- **Carga cognitiva**: En colaboración, el reparto de tareas o temas reduce la cantidad de procesamiento que debe realizar cada individuo.

- **Explicación**: Construir una explicación que debe ser dada a un grupo generalmente crea un conocimiento más solido.
- **Conflicto:** La discrepancia entre el conocimiento o el punto de vista de dos individuos lleva a la discusión, la cual genera que ambos individuos expliquen con detalle su punto de vista y escuchen el de sus pares, lo que beneficia a ambas partes.

Estos mecanismos de aprendizaje pueden verse reflejados en los efectos del aprendizaje colaborativo que se explican en la siguiente sección.

2.6 Efectos del aprendizaje colaborativo

La mayoría de las investigaciones en el trabajo colaborativo han intentado medir sus efectos generalmente a través de un examen individual antes y después de la actividad colaborativa [1]. Se deben de tomar en cuenta dos criterios al momento de realizar las mediciones de los efectos del aprendizaje colaborativo. El primer criterio está en definir que efecto en particular se va a medir, ya que una situación de aprendizaje colaborativo incluye e involucra una gran variedad de contextos e interacciones. El segundo criterio es el modo de evaluación. Los efectos del aprendizaje colaborativo son a menudo medidos de manera individual, es decir, por evaluaciones que miden el desempeño de un solo individuo. Se ha propuesto como una mejor evaluación la medición del desempeño del grupo, dentro de esta evaluación se debe de verificar si el desempeño del grupo ha aumentado de manera positiva o si los miembros del grupo han obtenido o mejorado sus habilidades para colaborar.

2.7 Beneficios de aprender y trabajar en equipo

Son diversos los beneficios que trae el trabajar y aprender de forma colaborativa, entre ellos se encuentran [3]

- Un grupo comprende mejor un problema que una sola persona.
- Existe responsabilidad compartida.
- Facilita la detección de errores.
- Un grupo posee más conocimiento que una sola persona.

- Habilita más alternativas para la resolución de problemas.
- Se produce sinergia: la efectividad y calidad de la producción de un grupo es mayor que la suma de lo que puede producir cada miembro en forma individual.
- Permite el acceso a un mayor volumen de información útil y filtrada debido a las contribuciones sintetizadas de otras personas.
- Agiliza los procesos de aprendizaje ante la posibilidad de recurrir a miembros experimentados del grupo.

2.8 Aprendizaje Colaborativo Asistido por Computadora

El campo del Aprendizaje Colaborativo Asistido por Computadora o de ahora en adelante CSCL (por sus siglas en ingles *Computer-Supported Collaborative Learning)* se encarga del análisis y diseño de tecnologías que permiten la colaboración entre personas para asistir los sistemas de aprendizaje que se encuentran en un salón de clases [4].

El CSCL resulta ser una herramienta muy útil para medir los efectos del aprendizaje colaborativo, ya que habilita un medio para grabar con detalle estas interacciones. Mediante el CSCL es posible ajustar los criterios y parámetros que se ven involucrados en el aprendizaje colaborativo con el fin de aumentar la probabilidad de que mediante las interacciones colaborativas se produzca un aprendizaje.

La meta general del CSCL es el desarrollo de productos y aplicaciones de software que le brinden a los usuarios actividades de exploración intelectual y de interacción social con el fin de aprender. El potencial creciente de internet para conectar a las personas de una manera más eficiente ha brindado un estímulo para la investigación en el CSCL. El creciente uso del CSCL ha ampliado las ventajas que se pueden obtener del software educativo, esto trae consigo cambios significativos en las instituciones, en los métodos de enseñanza y en el aprendizaje [5].

El uso de computadoras en el salón de clase se ha observado con escepticismo. Es visto generalmente como un mecanismo monótono y antisocial de enseñanza [6]. El CSCL está basado precisamente en la visión opuesta: intentar desarrollar nuevos escenarios y aplicaciones de software que brinden a los usuarios actividades didácticas y de interacción

en equipo. Las investigaciones recientes en el área del CSCL junto con el aumento de poder de procesamiento de las computadoras y las tecnologías de la comunicación están construyendo nuevas herramientas con el objetivo de diseñar escenarios que propicien el aprendizaje colaborativo. El emplear estos medios en el dominio del CSCL es considerado una oportunidad para la creación de dispositivos, actividades y entornos que ofrezcan el fortalecimiento de las prácticas de los usuarios en la construcción de conceptos y también como soporte a la evaluación de los mismos.

2.9 Ventajas del CSCL

En general las ventajas del aprendizaje colaborativo asistido por computadora son [7]:

- Genera una interdependencia positiva, los miembros del equipo se apoyan mutuamente y confían en la comprensión y conocimiento de cada persona.
- Promueve y ajusta las interacciones y el intercambio verbal entre las personas del grupo, lo que tiene efectos en los resultados del aprendizaje.
- Valora la contribución individual, debido a que cada integrante del equipo asume su responsabilidad en la tarea, a la vez que al exponerla recibe las contribuciones del grupo.
- Estimula las habilidades personales y de grupo al permitir que cada miembro participante desarrolle y potencie las habilidades personales y grupales como: escuchar, participar, liderar, coordinar actividades, realizar seguimiento y evaluar.

Además, otra gran ventaja de utilizar una computadora en lugar de otros medios (p. ej. lápiz y papel) para realizar actividades relacionadas con el aprendizaje colaborativo, es que se tiene cierto control de los parámetros y criterios que lo definen (situación, interacciones, aprendizaje, efectos) y con esto es posible hacer los ajustes necesarios para aumentar los beneficios del aprendizaje colaborativo.

CAPÍTULO 3

TECNOLOGÍAS WEB

Las aplicaciones web han tomado un gran auge en las últimas dos décadas a tal punto en que hacemos uso de ellas en diversas actividades cotidianas que van desde consultar un correo electrónico, hasta realizar una transferencia bancaria en el sitio web de un banco. Estas actividades se realizan a través de una red de computadoras (clientes y servidores) que intercambian información mediante distintos protocolo de comunicación, uno de los más comunes es el protocolo *http*.

Las tecnologías web también están revolucionando la manera en que aprendemos y enseñamos, esto mediante la creación de escenarios que hacen posible nuevas maneras de interactuar entre profesores y alumnos. Una de las áreas dentro de la educación que más se ha beneficiado es el aprendizaje colaborativo.

En las siguientes secciones se introducen los conceptos claves para comprender el funcionamiento de la aplicación web desarrollada en esta tesis (protocolo *http*, redes, arquitectura cliente-servidor, internet).

3.1 Protocolo *http*

El protocolo de transferencia de hipertexto o http (*HyperText Transfer Protocol*) es un protocolo cliente-servidor que establece como se debe de efectuar el intercambio de información entre los clientes web y los servidores. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como la *Word Wide Web* [8].

El protocolo se basa en operaciones de solicitud-respuesta, en el que un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. A su vez el servidor responde con un mensaje que contiene el estado de la operación y su resultado. Las operaciones pueden traer adjunto un objeto o recurso, cada objeto web (documento HTML, archivo multimedia, etc.) es localizado por su URL (*Uniform Resource Locators*). Este protocolo constituye la base de la *Word Wide Web,* que funciona mediante la implementación de URL para localizar datos en la Internet [9].

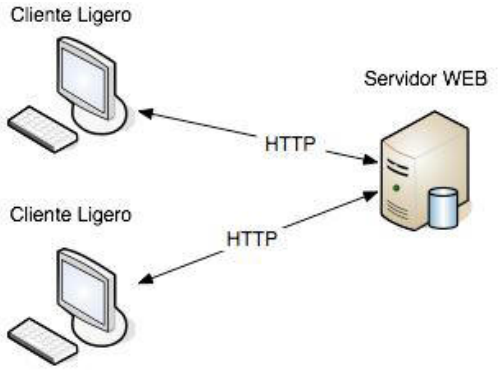


Figura 3.1.1 Esquema de comunicación entre un cliente y un servidor [10]

En la figura 3.1.1 se muestra el esquema de comunicación entre un cliente y un servidor, más adelante se explica lo que es un cliente ligero y las diferencias entre este y un cliente pesado.

3.2 Redes

Una red es un conjunto de computadoras conectadas entre sí para compartir recursos. Las redes se pueden clasificar en varias categorías siendo las más reconocidas: las redes de área local (LAN, *Local Area Network*) y las redes de área amplia (WAN, *Wide Area Network*). Una Red de Área Local permite a las computadoras situadas en un espacio físico relativamente limitado (por ejemplo, una casa, una oficina, un par de pisos en un edificio) acceder a recursos compartidos como impresoras, documentos digitales, o computadoras más potentes llamadas servidores. Por su parte, una WAN es una red que enlaza computadoras que se encuentran en diferentes redes de área local en una zona geográfica amplia, es decir, conecta diferentes redes LAN creando una red más amplia y permitiendo así un mayor número de computadoras conectadas. La red WAN más conocida y popular en la actualidad es la *Word Wide Web* [9].

3.3 Internet y *Word Wide Web*

La *World Wide Web* fue creada en 1989 en el CERN (*European Laboratory for Paricles Physics*). Su difusión masiva comenzó en 1993 como un medio de comunicación universal entre redes de computadoras. La web es un sistema de estándares aceptados

universalmente para almacenar, recuperar y visualizar información utilizando una arquitectura cliente-servidor. Todos los datos, incluyendo mensajes de correo electrónico y las páginas web, se almacenan en servidores. Un cliente (usuario) utiliza internet para solicitar información de un servidor web determinado situado en una computadora normalmente lejana, el servidor envía la información solicitada al cliente vía Internet.

La web ha cambiado el mundo, se ha convertido sin dudas en el medio de comunicación más poderoso que se haya conocido. Este medio de comunicación ha cambiado la forma en que enseñamos, aprendemos, compramos, vendemos, nos informamos, compartimos, conocemos y sus alcances han tenido efecto en la manera en que convivimos con otras personas [11].

3.4 Arquitectura cliente – servidor

Una aplicación web se puede modelar como un conjunto de servicios proporcionados por servidores y un conjunto de clientes que usan estos servicios. En la figura 3.4.1 se muestra una arquitectura cliente-servidor, donde los servidores están representados con la letra “S” y los clientes con la letra “C”.

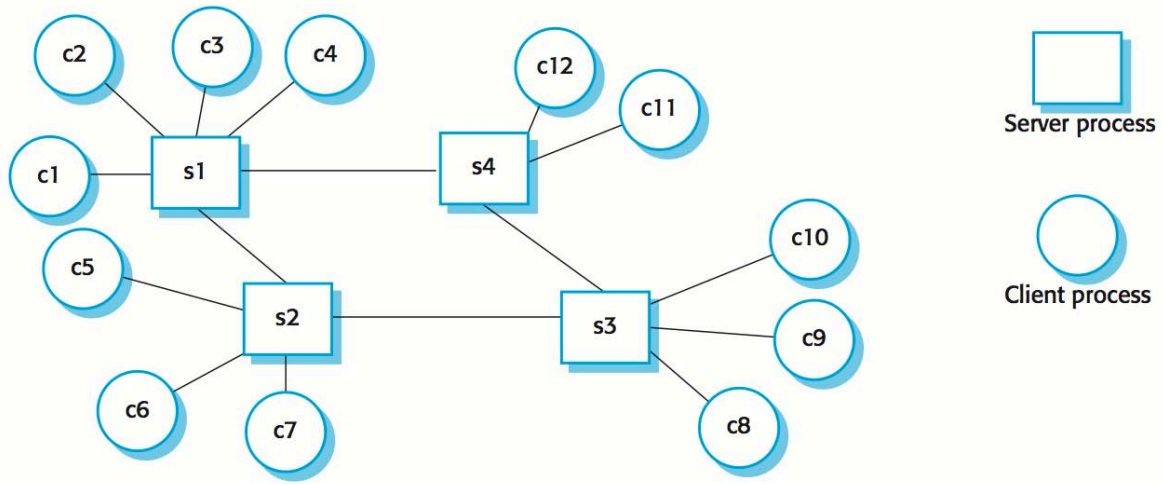


Figura 3.4.1 Un sistema cliente-servidor [12]

La arquitectura cliente-servidor más simple se denomina “arquitectura cliente-servidor de dos capas”, y es en la que una aplicación se organiza como un servidor y un conjunto de clientes. Las arquitecturas cliente-servidor de dos capas pueden ser de dos tipos:

- **Modelo de cliente ligero.** En un modelo de cliente ligero todo el procesamiento de las aplicaciones y la gestión de los datos se lleva a cabo en el servidor. El cliente solamente es responsable de la capa de presentación del interfaz de usuario.
- **Modelo de cliente pesado**. En este modelo, el servidor es responsable únicamente de la gestión de los datos. El software del cliente implementa la lógica de la aplicación y las interacciones con el usuario del sistema.

En nuestro caso, el desarrollo de la aplicación para el aprendizaje colaborativo se basa en la arquitectura cliente pesado–servidor; pues el usuario (cliente) se encarga de procesar y enviar la información al servidor. El desarrollo del software para el funcionamiento de la aplicación se divide en dos partes: software del lado del cliente y software del lado servidor. En cada parte se utilizan tecnologías diferentes, estas se presentan a continuación.

3.5 Tecnologías web del lado del cliente

Comenzaremos describiendo las tecnologías empleadas en esta tesis en el lado del cliente (o *front-end*). Como ya se mencionó anteriormente, la aplicación que estaremos desarrollando en este trabajo cae dentro de la categoría de cliente pesado, lo que involucra herramientas para presentar, procesar y comunicar la información desde el cliente al servidor, así como recibir información desde el servidor al cliente y procesarla. En una aplicación web cada uno de estos procesos requiere de un código particular: HTML para dar estructura a la interfaz gráfica del usuario, CSS para dar estilo y JavaScript para hacer dinámica la aplicación.

HTML5

HTML5 es un estándar para el desarrollo de aplicaciones web que ofrece una amplia gama de funcionalidades. Versiones anteriores de HTML estaban centradas principalmente en el concepto de un lenguaje de marcado estático para documentos,

enfocado únicamente a dar estructura a la interfaz gráfica del usuario en páginas web [13]. HTML5 es la primera versión que propone un nuevo estándar para el desarrollo de aplicaciones web, es decir, permite ver al lenguaje de marcado de hipertexto como una plataforma de desarrollo.

Esta nueva versión de HTML define elementos que pueden ser usados para el desarrollo de aplicaciones de internet enriquecidas (reproducción de videos de manera nativa, *web Sockets*, elementos visuales más versátiles), así como una gran variedad de estándares JavaScript. Entre las ventajas que ofrece HTML5 se encuentran: la creación de juegos, construir aplicaciones que se visualicen en múltiples pantallas como en móviles, tabletas y laptops, entre muchas funcionalidades más. Además, el hecho de que se trate de un estándar abierto ha permitido que se hagan implementaciones en diversos navegadores y plataformas [14].

CSS y Bootstrap

Una vez que se ha definido la estructura visual de una aplicación web mediante HTML, sigue agregar características para hacerla más atractiva a los usuarios, de esto se encarga CSS. Las hojas de estilo en cascada o CSS (*Cascading Style Sheets*) son un lenguaje de estilo introducido por el consorcio *Word Wide Web*. El término "en cascada" se refiere al proceso de determinar la prioridad de las reglas de estilo (apariencia de los elementos HTML) en un sitio web.

CSS provee control sobre las características visuales en documentos HTML y sus elementos. Algunas características típicas que se controlan a través de documentos CSS son, por ejemplo: fuente, colores, fondos, márgenes, bordes y capas. Además CSS provee poderosas características para soportar más de un medio visual y alcanzar navegadores especiales como los que se encuentran en los diferentes dispositivos móviles [15]

Por otro lado, existen librerías que permiten aprovechar de manera más efectiva las funcionalidades de CSS, una de ella es *Bootstrap,* la cual provee un conjunto de clases CSS y funciones JavaScript para facilitar los procesos de desarrollo en *front-end* [16]. Desde su aparición en agosto del 2011 de *Twitter Boostrap* (ahora simplemente *Bootstrap)*, se ha convertido en el marco de trabajo más popular para el diseño web [17]. La

popularidad de *Bootstrap* en el desarrollo web se debe en gran medida a que provee una amplio rango de soluciones para la mayoría de los estándares de interfaz de usuario que son amigables con el desarrollador y que funcionan en la mayoría de los exploradores.

JavaScript

Desde los primeros días de la Web ha existido la necesidad de crear interfaces que proporcionen una mejor experiencia del usuario. En un inicio las páginas web eran estáticas (solo se mostraba texto plano), pero conforme creció la popularidad y tamaño de la web, también creció la necesidad de crear interacciones más versátiles con el usuario. En respuesta a ello fue como nació JavaScript.

JavaScript es un lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo [18]. Es importante aclarar que JavaScript no es "Java interpretativo". JavaScript es un lenguaje de programación dinámico que soporta construcción de objetos basados en prototipos. Su sintaxis es similar a Java y C++ con la intención de reducir el número de conceptos nuevos para aprender el lenguaje. Las sentencias y ciclos de control del lenguaje, tales como *if*, y bucles *for* y *while*, y bloques *switch* y *try ... catch* funcionan de la misma manera que en estos lenguajes (Java y C++).

Una de las grandes ventajas es que JavaScript puede funcionar como lenguaje procedimental y como lenguaje orientado a objetos. Los objetos se crean añadiendo métodos y propiedades a lo que de otra forma serían objetos vacíos en tiempo de ejecución, contrario a las definiciones sintácticas de clases comunes en los lenguajes compilados como C++ y Java. Una vez que se ha construido un objeto, puede usarse como prototipo para crear objetos similares.

Entre las capacidades de JavaScript se encuentran la construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de *scripts* dinámicos, introspección de objetos y recuperación de código fuente.

Por último, otra gran ventaja de emplear JavaScript es que sus objetos son representados mediante un tipo de dato muy versátil y ampliamente usado en la actualidad llamado JSON.

JSON

JavaScript Object Notation es un formato ligero para el intercambio de información. Es fácil de leer y escribir para los humanos y sencillo de generar y analizar para las máquinas. Está basado en el lenguaje JavaScript. Además es un formato de texto que es completamente compartible con cualquier lenguaje y utiliza convenciones que son familiares para programadores de la familia del lenguaje C, incluyendo C++, C#, Java, JavaScript, Perl, Python, entre otros. Estas propiedades hacen a JSON un lenguaje ideal para el intercambio de datos [19].

JSON está construido con dos estructuras:

- **Una colección de pares: nombres y valores**. En varios lenguajes, esto es conocido como objeto, record, estructura, diccionario, tabla *hash*, lista con llaves o arreglos asociados.
- **Una lista ordenada de valores.** En la mayoría de los lenguajes, esto es conocido como arreglo, vector, lista o secuencia.

Estas son estructuras de datos universales. Virtualmente todos los lenguajes de programación modernos soportan JSON de alguna manera. En JSON un objeto es un conjunto de pares nombres/valores. Un objeto comienza con "{" y termina con "}". Cada nombre es seguido por ":" y el valor, y el par nombre-valor se separa por una coma ",". En la figura 3.5.1 se muestran los elementos que componen a un objeto JSON.

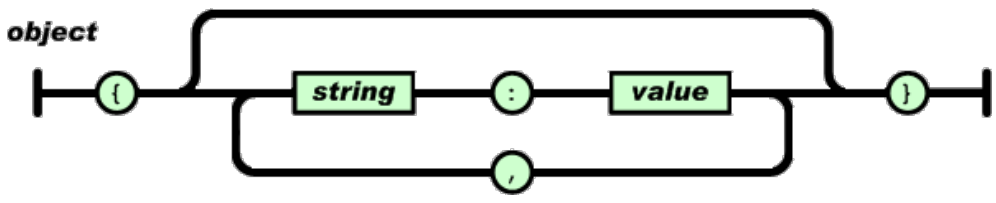


Figura 3.5.1 Estructura de un JSON

JQuery

jQuery es una librería de JavaScript diseñada para reducir los tiempos de desarrollo del lado del cliente ya que provee un conjunto de funciones de propósito general que permiten simplificar tareas durante la construcción de sitios web interactivos como la aplicación desarrollada en esta tesis. A continuación se listan las principales bondades que brinda jQuery:

- **Acceso a los elementos del DOM** (*Document Object Model*): Sin una librería de JavaScript a menudo es necesario escribir varias líneas de código para acceder a los elementos del DOM y localizar recursos en una estructura de un documento HTML. jQuery provee un mecanismo de selección robusto y eficiente, haciendo más fácil acceder a un elemento determinado que se necesita manipular.
- **Modificar la apariencia de páginas web:** con jQuery es posible cambiar las clases o propiedades de estilos individuales aplicados a un elemento del documento incluso después de que la página ha sido desplegada inicialmente en el navegador.
- **Modificar el contenido de un documento**: mediante la librería jQuery se puede modificar el contenido de un documento con pocas líneas. El texto puede ser cambiado, imágenes pueden ser insertadas o eliminadas, las listas pueden ser ordenadas o la estructura entera de HTML puede ser reescrita o extendida.
- **Responder a las interacciones de usuario**: jQuery ofrece métodos para detectar y procesar una amplia variedad de eventos, tales como hacer un clic en una liga, sin la necesidad de introducir manejadores de eventos en el código HTML.
- **Solicitar información a un servidor:** mediante AJAX (*Asynchronous JavaScript and XML*) es posible hacer una petición al servidor sin necesidad de refrescar la página simplificando la comunicación entre clientes y servidores.

Estas ventajas convierten a JQuery en la librería más usada de JavaScript hoy en día [20] Ahora pasaremos a introducir las tecnologías web empleadas en el lado del servidor.

3.6 Tecnologías web del lado del servidor

Existen diversos lenguajes y marcos de trabajo para el desarrollo de aplicaciones web del lado del servidor, entre los más comunes se encuentran: Java, PHP, C#, Python, Ruby, y así como todos sus *frameworks* (*Spring, .NET, Django, Ruby on Rails, etc.*). Es importante señalar que cada opción cuenta con virtudes que se adecuan de diferente forma al problema a solucionar. La aplicación a desarrollar en esta tesis demanda una arquitectura que soporte transmisión de datos en tiempo real y una concurrencia de múltiples a usuarios a la vez. Partiendo de estos requisitos, se encontró que una de las soluciones que más se adecua es emplear Node.js (servidor web que emplea Javascript) ya que fue diseñado para trabajar eficientemente con la transferencia de datos en tiempo real y con una alta concurrencia. Además existen módulos para Node.js que simplifican y diversifican las aplicaciones que se pueden desarrollar. En esta tesis se empleó Express (librería de Node.js) como marco de trabajo web y Socket.io (librería de node.js) para cubrir la transferencia de datos en tiempo real. A continuación se presenta una introducción a Node.js, Express y Socket.io.

Node.Js

En febrero del 2009, Ryan Gahl tuvo la visión de crear una combinación entre JavaScript y el motor V8 de Google para crear una nueva plataforma de programación. Así creo Node.js y lo publicó el mismo mes. Node.js fue muy bien recibido por la comunidad del desarrollo web y su popularidad comenzó a crecer muy rápidamente. Se trata de una plataforma empleada como servidor, basada en la máquina de ejecución de *Google Chrome*, diseñada para construir aplicaciones en red escalables, multiplataforma y robustas de manera rápida y fácil, además de estar optimizada para emplearse en aplicaciones que requieren de una intensa transferencia de datos en tiempo real [21].

Node.js provee una API (*Application Programming Interface*) con el propósito de brindar a los desarrolladores herramientas para crear aplicaciones web usando JavaScript como lenguaje de programación en el *Back-end*. Quizá la única desventaja con la API del servidor *http* que ofrece Node.js es que es de bajo nivel, por lo que se tiene que escribir muchas de las funciones de un servidor *http* en la aplicación web. La creación de módulos y la extensibilidad se convierte en un problema para cualquier proyecto que es

moderadamente grande. Es por ello que nació la necesidad de crear un marco de trabajo que permitiera hacer el desarrollo web un poco más fácil en Node.js. Bajo esta necesidad se creó Express.

Express

Es un marco de trabajo de desarrollo web flexible y de gran alcance para la plataforma Node.js [22]. Express es ligero ya que solo soporta los aspectos básicos del desarrollo web. Es flexible porque está integrado con el uso de *Middlewares* (lógica de intercambio de información entre aplicaciones) y módulos de Node.js. El *middleware* de Express y los módulos de Node.js son componentes compartibles mediante JavaScript, lo que hacen de las aplicaciones desarrolladas con Express sean modulares y extensibles. Express brinda un acceso completo al núcleo de la API de Node.js, lo que lo convierte en un marco de trabajo versátil (p. ej. todo lo que se pueda hacer en Node.js, se puede hacer con Express).

Express fue inspirado por *Sinatra*, un marco de trabajo de Ruby y construido sobre el API del servidor web Node.js. Además provee rutas al sistema, soporte y manejo de *cookies*, módulos MIME (*Multipurpose Internet Mail Extensions*), una interfaz para servicios web, entre otras cosas.

Socket IO

Socket.IO es una librería escrita en JavaScript que permite la comunicación bidireccional, en tiempo real y basada en eventos entre varios clientes y varios servidores. Funciona en cualquier plataforma, navegador o dispositivo, concentrándose en el equilibrio entre velocidad y fiabilidad [23]

Una aplicación en tiempo real puede recibir alertas o mensajes nuevos sin que el usuario haya ejecutado alguna acción. Las aplicaciones web en tiempo real son un conjunto de tecnologías y técnicas que permiten al usuario tener acceso a información tan rápido como sea emitida por el emisor sin necesidad de hacer peticiones periódicas a esta información. Este conjunto de tecnologías son ampliamente usadas en la actualidad en aplicaciones que

requieren actualización de datos en tiempo real, algunos ejemplos son aplicaciones relacionadas con el mercado de valores, divisas, el clima, geo localización y redes sociales. Debido a la importancia que tienen los *web sockets* en esta tesis la siguiente sección se dedica a dar una breve introducción a su nacimiento.

3.7 Evolución de los web sockets

El desarrollo de aplicaciones que intercambian y procesan información en tiempo real no es un tema nuevo. Uno de los primeros intentos en hacer aplicaciones con estas características fue a través del uso de *applets* de Java. Hoy en día ha cambiado la manera en que estas tecnologías funcionan, reduciendo los costos de procesamiento y recursos, y se han convertido en una necesidad en muchas de las aplicaciones web actuales. Estas tecnologías se conocen comúnmente como *WebSockets* y Eventos Mandados por el Servidor (*Server-Sent Events*, *SSE*).

Como se aprecia en la figura 3.7.1 las aplicaciones web que se ejecutan en un navegador están basadas en protocolos http, que son empleadas por sistemas petición-respuesta en el que el cliente manda una petición de información a el servidor y este responde con la información solicitada. En la mayoría de los casos esta información es un archivo HTML, XML, JSON o algo relacionado para visualizar información en el navegador.

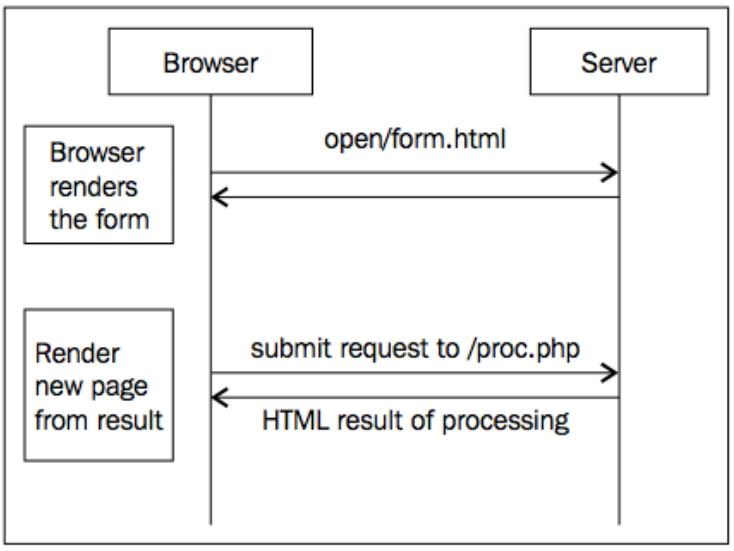


Figura 3.7.1 Interacción HTTP entre cliente y servidor [24]

Con el propósito de crear interfaces más interactivas, en 1995 nacieron los *applets* de Java. Muchos sitios web los integraron para permitir la implementación de aplicaciones web que reaccionaban en tiempo real, como chats, juegos o anuncios. En el mismo año, Netscape creo un lenguaje de *script* llamado JavaScript, mismo que se considera la causa de desaparición de los *applets* de la mayoría de las páginas web. En 1999, Microsoft implementó su propia tecnología para actualizar noticias relacionadas con el mercado de valores en Internet Explorer lanzando la tecnología llamada XMLHTTP, la cual fue un gran avance en el intercambio de información entre un cliente y un servidor.

Este componente, XMLHTTP, fue originalmente creado para cargar información XML en una página web de manera asincrónica, usando JavaScript. Fue rápidamente adoptado por Mozilla, Safari y Opera como XMLHttpRequest (XHR). En la figura 3.7.2 se muestra un esquema del funcionamiento de los XHR, haciendo uso de estos es posible hacer peticiones al servidor sin la necesidad de refrescar la página lo que permite actualizar información en el cliente en cualquier momento.

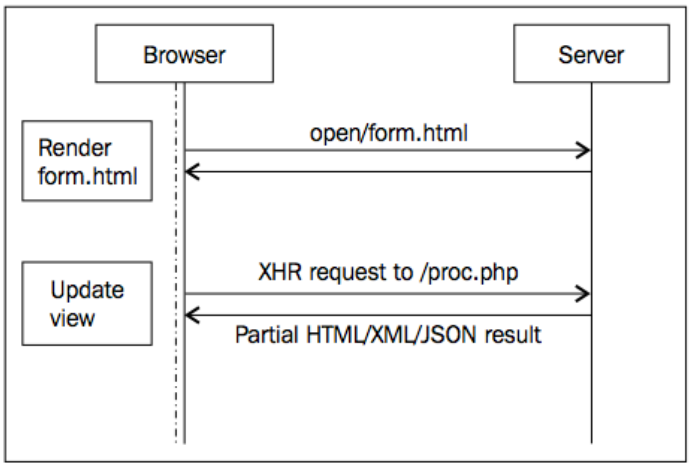


Figura 3.7.2 Petición XHR [24]

Con el lanzamiento de *Gmail*, Google introdujo el término AJAX, que en cierta manera es una variante de XHR. AJAX ganó más popularidad que su predecesor entre los desarrolladores debido principalmente a su simplicidad. Desde la introducción de peticiones asincrónicas, fueron evidentes las ventajas de actualizar información en vivo en páginas web. Colectivamente, estas tecnologías fueron referidas como *Comet*, un término introducido por Alex Russel[24]. *Comet* introdujo mecanismos que dan la impresión de

que los datos son enviados desde el servidor al cliente, sin que el cliente los haya solicitado explícitamente. Comet incluye: *Hidden iframe*, XHR *polling*, XHR *long polling*, y *Script tag long pilling* (o JSONP *long polling*).

Estas técnicas son actualmente los mecanismos más comunes disponibles en los navegadores modernos. El primero y más fácil de implementar es el XHR *polling*. Este se muestra en la figura 3.7.3 y ocurre cuando el navegador solicita información al servidor de manera periódica y el servidor responde con información vacía hasta que haya datos que regresar al cliente. Una vez que el servidor tienen datos que enviar, como un correo, o actualizar un registro en una base de datos, el servidor responde a la siguiente petición con estos nuevos datos.

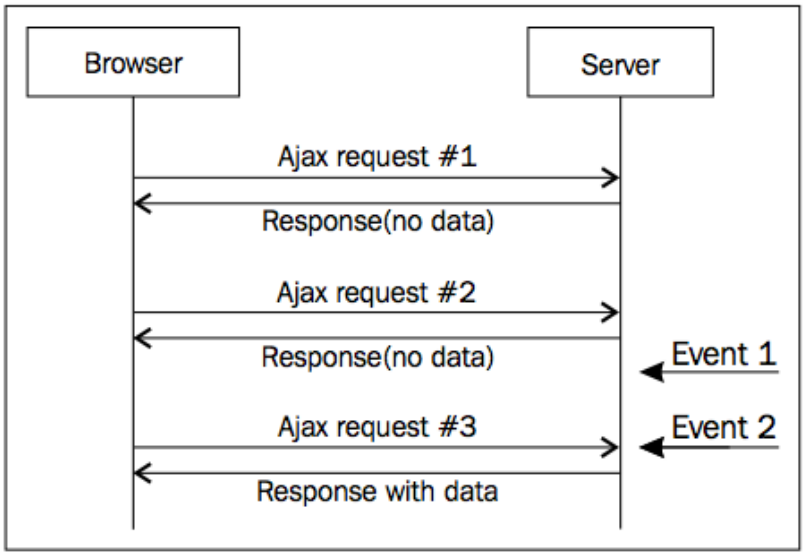


Figura 3.7.3 XHR *polling*. [24]

El problema de esta técnica es que el cliente tiene que hacer peticiones al servidor aún cuando no hay datos que entregar. Esto causa que el servidor tenga que procesar las peticiones provenientes del cliente, lo que se vuelve muy ineficiente en el consumo de recursos (procesamiento y ancho de banda de red).

Una variación a este mecanismo, que busca disminuir el número de mensajes enviados, es una técnica llamada *long polling*. En *long polling* (figura 3.7.4) cuando el cliente manda una petición al servidor, este no responde inmediatamente hasta que cuente con datos que transmitir, y suspende temporalmente la respuesta. Una vez que un nuevo evento ocurre, el

servidor concluye la solicitud suspendida mandando la respuesta al cliente, tan rápido como el cliente reciba la respuesta, mandará una nueva petición.

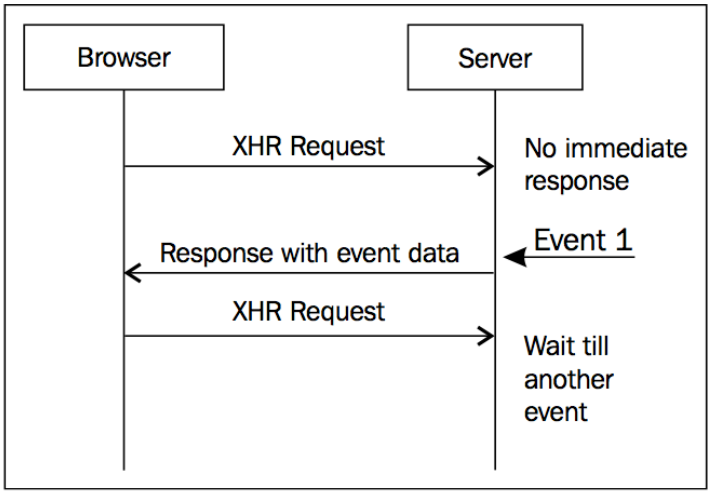


Figura 3.7.4 En la técnica *Long polling* el servidor no responde al cliente hasta que exista información relevante que enviar [24]

Hay varias tecnologías en las que el *long polling* es implementado, como en el *forever iframe*, *multipart* HXR, *script tags* con JSONP y XHR de larga vida. Aunque todas estas técnicas funcionan, se consideran parches, ya que se aprovechan de HTTP y XHR para que sean capaces de establecer una comunicación bidireccional, algo para lo que no fueron hechos.

HTML5 brinda dos métodos nuevos para enviar datos del servidor al cliente, uno es con *Server-Sent Events* (SSE) y el otro es con *WebSockets*.

Server-Sent Events intenta estandarizar la comunicación bidireccional a través de todos los navegadores. Esta técnica cuenta con su propia API de JavaScript. En los *WebSockets*, el cliente inicia la conexión con el servidor, el cual soporta el protocolo también. El servidor y el cliente se envían y reciben información mediante este canal establecido.

Una vez introducidas las tecnologías web empleadas en esta tesis, así como su funcionamiento, pasaremos a explicar como se desarrollaron las partes que componen a la aplicación.

CAPÍTULO 4

DISEÑO Y DESARROLLO DE FRONT-END

El diseño y desarrollo de una aplicación web se puede dividir en dos partes:

- El lado del cliente o *front-end* que corresponde al código de la interfaz gráfica con el que interactúa el usuario.
- El lado del servidor o *back-end* que corresponde al código que se encuentra en el servidor que se encarga de sincronizar y recibir las peticiones http de los clientes.

En este capítulo se describe el funcionamiento y los requisitos que debe cubrir la aplicación, así como del diseño y desarrollo en el lado del cliente.

4.1 Descripción de la aplicación web "Mural Colaborativo"

La aplicación web desarrollada en este trabajo busca mostrar a través de un ejemplo concreto que las tecnologías y arquitecturas propuestas funciona de manera eficiente y dan respuesta a las necesidades del proyecto "El Aula del Futuro". Estas tecnologías deben soportar una aplicación multiusuario en la que se sincroniza y comparte el trabajo individual con un área de trabajo pública común. Para ello, junto con los investigadores del CCADET se establecieron los siguientes requerimientos:

- Desarrollar una nueva versión del Escritorio Colaborativo que, inspirado en los trabajos previos del "Aula del Futuro", permita a un grupo de usuarios colaborar en la solución de una tarea. Para ello, cada usuario debe poder compartir el contenido generado en su dispositivo móvil en un espacio público compartido, a través del simple gesto de arrastrar la información de lo privado a lo público.
- La aplicación deberá estar desarrollada en lenguajes y estándares abiertos.
- La nueva versión del Escritorio Colaborativo deberá considerar: La aplicación del usuario en su dispositivo móvil, La aplicación del espacio público compartido y el servidor necesario para sincronizar la información que se intercambia entre los diferentes clientes.

Si bien estos requerimientos permitieron elaborar un primer análisis sobre la tecnología a utilizar y con ello desarrollar una propuesta sobre la arquitectura requerida, fue necesario definir un editor colaborativo concreto que permitiera definir las interfaces de usuario y los

servicios cliente-servidor asociados a programar. Así surge la idea de crear el “Mural Colaborativo”, un editor inédito en el laboratorio que conjunta el trabajo lógico-matemático con aspectos lúdicos, lo que le brinda el potencial educativo, así como de desarrollo de habilidades digitales y de colaboración que se promueven en el laboratorio.

La idea inicial del mural colaborativo es que en la parte pública se proyectan imágenes generadas a partir de mosaicos cuadrados, que están divididos en 36 triángulos (figura 4.1.1). Estas imágenes deben ser reproducidas por el grupo de usuarios. Para ello, cada uno debe de crear en la zona privada los mosaicos que acuerde el equipo y que permitan reproducir la imagen.

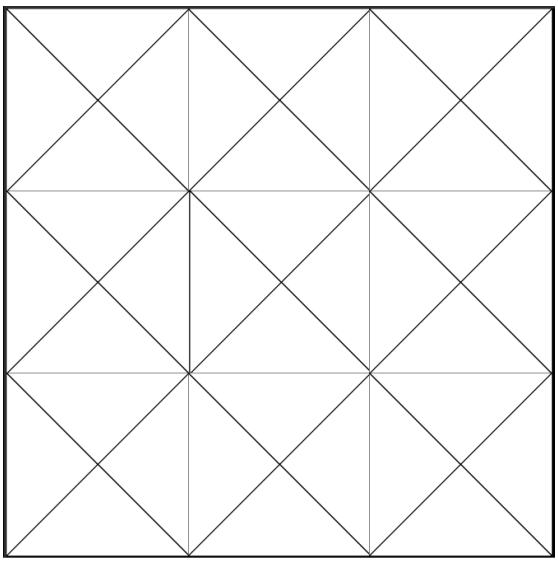


Figura 4.1.1 Imagen de un mosaico y sus divisiones.

Así, a partir de los mosaicos generados individualmente la imagen que se encuentra en la zona pública (figura 4.1.2) se es replicada. Este tipo de ejercicios permite a los alumnos desarrollar habilidades espaciales (dirección y rotación de figuras), de abstracción (generación de patrones) y lógico-matemáticas (fracciones).

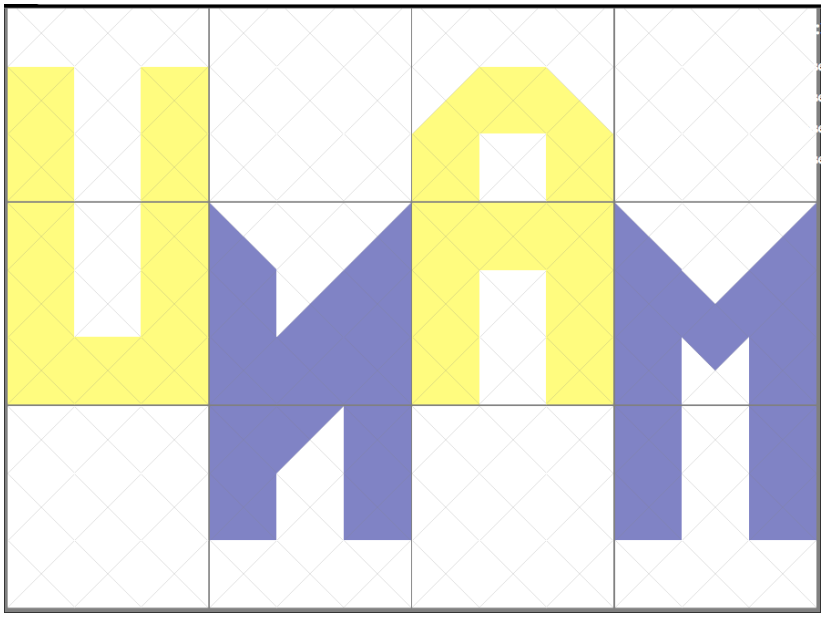


Figura 4.1.2 Imagen en la zona pública de un mural a completar con mosaicos

Con estas ideas en mente, la aplicación se compone de una interfaz web que muestra a un grupo de alumnos un mural que conforma una figura y el objetivo de la actividad consiste en que el grupo de alumnos trabaje colaborativamente construyendo las partes del mural con el fin de completarlo.

Como se muestra en la siguiente figura (4.1.3), el mural se presenta en una zona pública, y el grupo de alumnos debe planear colaborativamente la solución para replicar el mural en conjunto, pero trabajarán individualmente en una zona privada para crear la sección o fragmento que le corresponde a cada uno. En esta versión pueden participar desde uno a seis usuarios

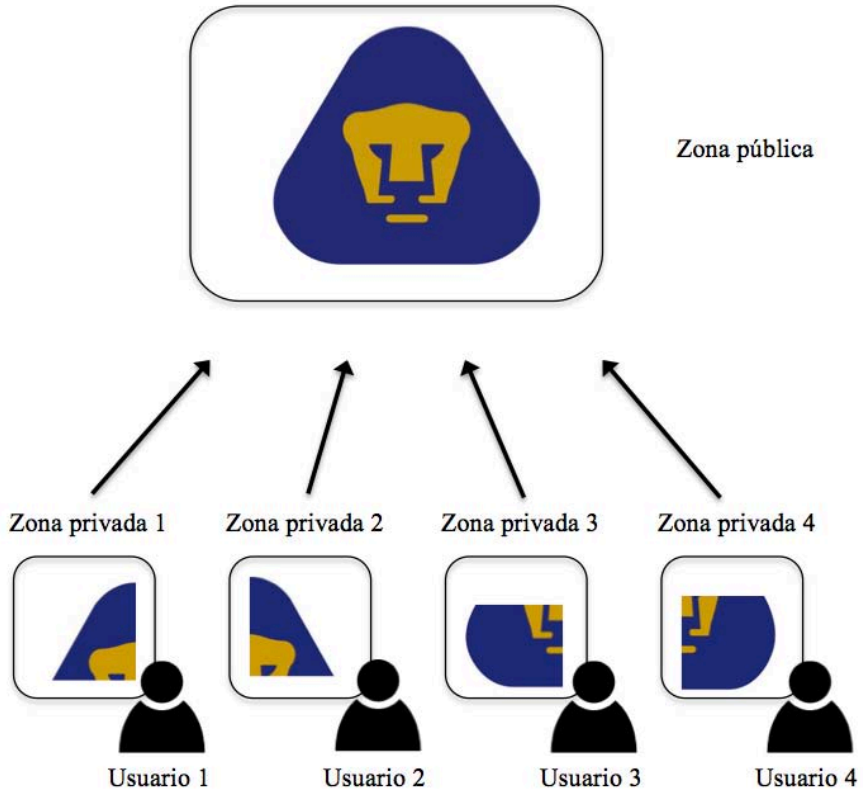


Figura 4.1.3 Esquema conceptual de la aplicación web a desarrollar

La zona pública debe ser un lugar visualmente accesible a todos, como por ejemplo una proyección en una pared, una televisión, el monitor de una computadora, etc. Por otro lado, como ya se mencionó en los requerimientos iniciales, la zona privada donde trabaja individualmente cada usuario puede ser una computadora, una tableta electrónica o un celular, prácticamente cualquier dispositivo que cuente con un navegador web.

4.2 Zona privada de trabajo

Se definieron inicialmente las tareas base que un usuario debe realizar en el área privada de trabajo:

- Crear piezas o mosaicos.
- Ver las piezas que ha generado y seleccionarlas.
- Trasladar las piezas generadas al área pública y ubicarlas sin ambigüedades en el lugar donde las deja.

- Editar alguna de las piezas hechas anteriormente.

Una vez establecido el funcionamiento y los requisitos de la aplicación pasamos a describir los pasos y herramientas que se emplearon para desarrollar la parte del *front-end* de la aplicación.

4.3 Diseño de la zona pública y la zona privada

Antes de comenzar a desarrollar o escribir código es importante diseñar y dejar en claro que funcionalidades debe de cubrir la interfaz gráfica de la aplicación, para después hacer esquemas o prototipos en herramientas especializadas que permiten una edición visual rápida de la interfaz de usuario. Con esto en mente, se representó la propuesta inicial de interfaz gráfica para la zona privada de la aplicación en un prototipo de diseño o *mock-up* mediante la aplicación www.moqups.com. Este prototipo de diseño permite en un estado inicial de la aplicación la adquisición de comentarios por parte de los usuarios y evaluadores. Esto facilita la edición del diseño, ya que mediante la herramienta *moqups* es más sencillo agregar, quitar o mover elementos que hacerlo mediante código HTML. A continuación se muestra la propuesta inicial de la zona privada de la aplicación.

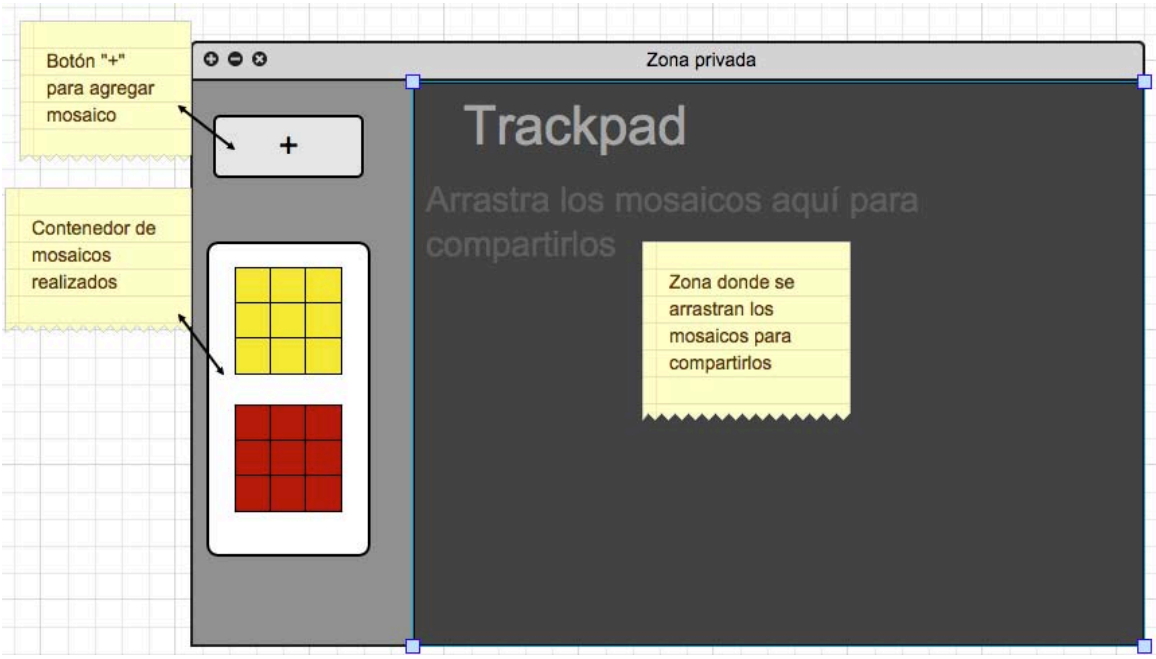


Figura 4.2.1 Prototipo de pantalla inicial en zona privada

En la interfaz el usuario (figura 4.2.1) se pueden realizar principalmente tres tareas:

1). Elaborar una sección del mural, que de ahora en adelante llamaremos mosaico. Esto se hace mediante el botón “+” que se encuentra en la parte izquierda de la pantalla.

2). Editar mosaicos que se realizaron previamente y que se encuentran en el contenedor de mosaicos que está posicionado en la parte izquierda de la pantalla. Para editar un mosaico el usuario debe hacer doble clic o doble *tap* en el mosaico a editar.

3). Compartir un mosaico. Para compartir un mosaico se debe arrastrar este desde el contenedor de mosaicos al *trackpad.* Una vez en el *trackpad* el usuario puede ver sus movimientos reflejados en la zona pública y deberá “soltar” (o dejar de arrastrar) el mosaico en la posición que el usuario desee imprimir el mosaico.

A continuación se presenta la interfaz que se muestra al usuario cuando este presiona el botón “+”

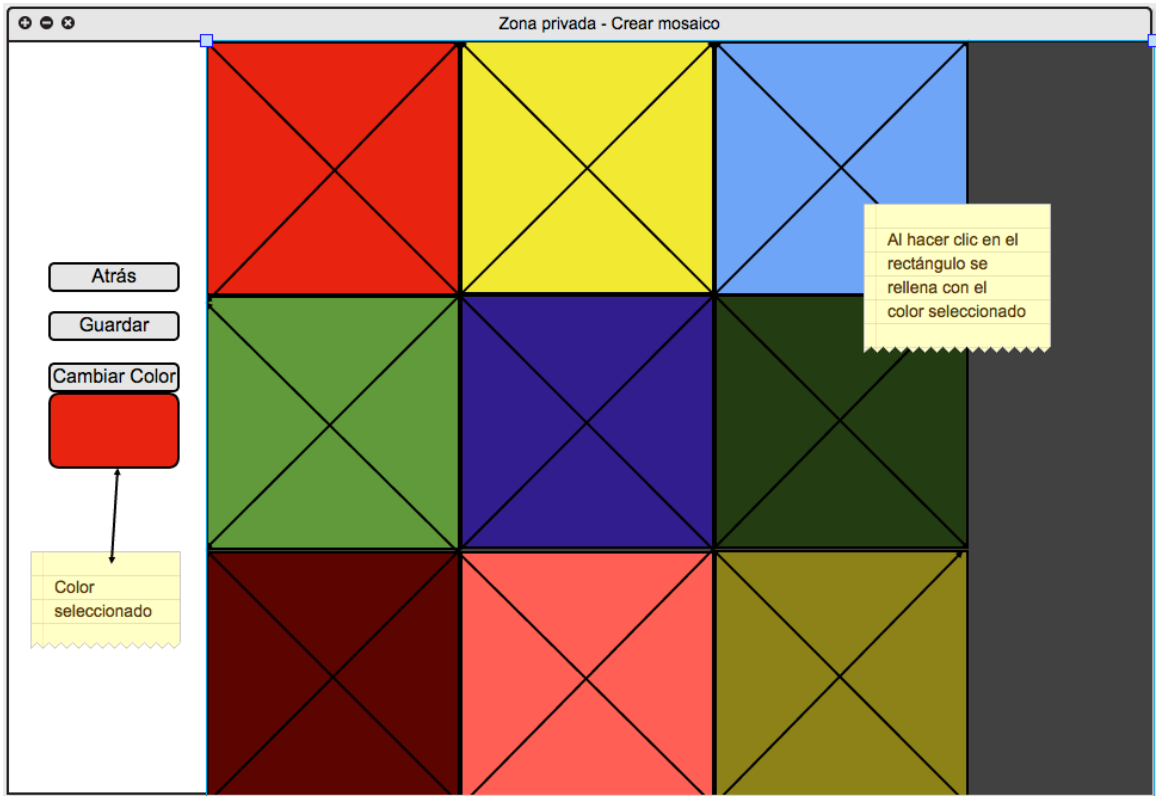


Figura 4.2.2 Prototipo de interfaz para agregar mosaicos

Una vez que el usuario presiona el botón de agregar o “+” la aplicación se dirige a la interfaz donde se elaboran los mosaicos (figura 4.2.2). En esta interfaz se realizan cuatro actividades:

1). Regresar al interfaz inicial con el botón “atrás” sin guardar el mosaico.

2). Elegir el color a emplear en una sección del mosaico (triángulos) mediante el botón paleta de colores, donde se indica que color está seleccionado actualmente.

3). Definir el color o “colorear” cada triángulo del mosaico haciendo clic o *tap* en el mismo.

4). Guardar el mosaico mediante el botón de “guardar” una vez que el usuario considere terminada la elaboración del mosaico.

Ya que el usuario ha elaborado un mosaico y lo comparte arrastrándolo en el *trackpad* a la zona pública, este se visualizará en el mural (fig. 4.2.3) en la posición donde lo soltó.

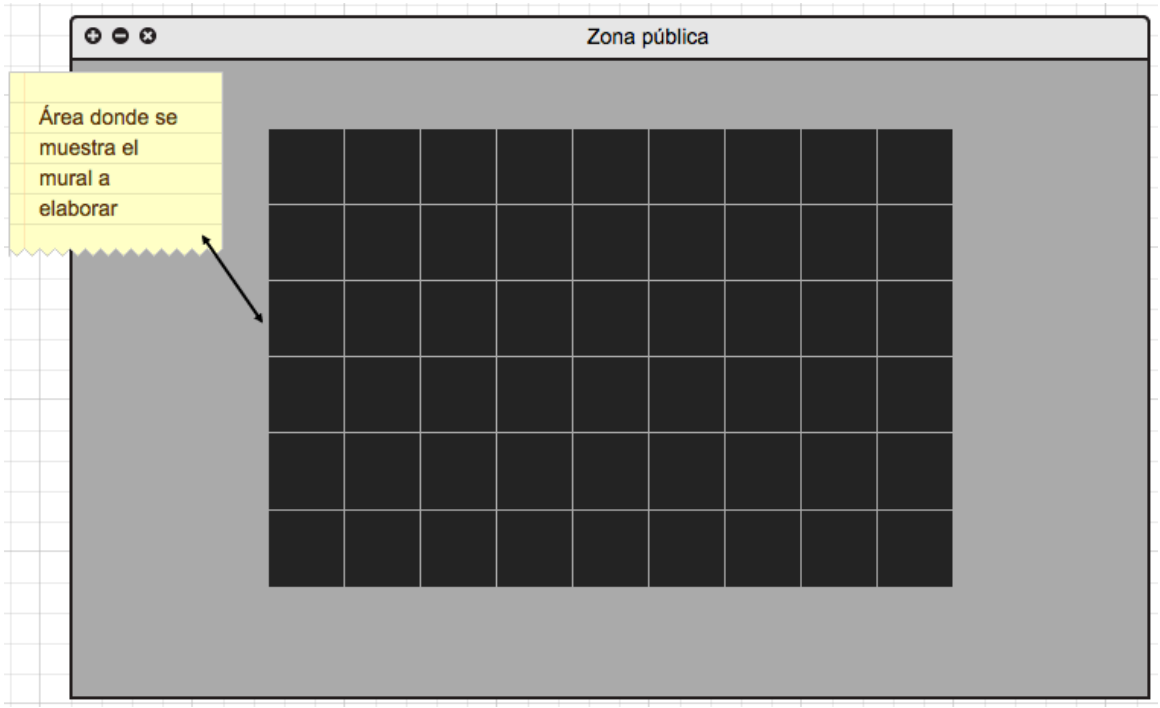


Figura 4.2.3 Prototipo de diseño de la zona pública

Una vez definido los prototipos de la interfaz gráfica procedimos a desarrollar el código y arquitectura necesaria para su funcionamiento.

4.4 Desarrollo de Front-end

Para el desarrollo del *front-end* de la aplicación se empleó principalmente HTML5 para dar la estructura, CSS3 para dar estilo y JavaScript para hacer dinámicos los elementos mediante las librerías *JQuery, BootStrap, SocketIO.*

El proyecto (donde se encuentra todo el código) se creó mediante Express. Express es un marco de trabajo para el desarrollo de aplicaciones web que se encarga de elaborar la estructura del proyecto. El objetivo de escribir en este capítulo como se encuentra distribuido el código y señalar las funciones esenciales para el funcionamiento de la aplicación es con el fin de guiar a futuros desarrolladores interesados en continuar desarrollando esta aplicación y dar al lector en general una idea de cómo funcionan las tecnologías empleadas.

En la carpeta *public* que se muestra en la figura 4.4.1 se encuentran todos los archivos que se ejecutan en el lado del cliente. En la carpeta *view* se encuentran los archivos HTML con extensión .ejs (*embedded JavaScript*) que contienen la interfaz gráfica que verá el usuario

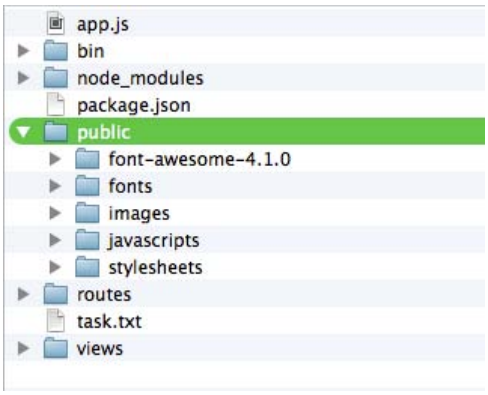


Figura 4.4.1 Estructura del proyecto

Se utilizó el paradigma de programación dirigida por eventos en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurren en el sistema y son definidos y provocados por el usuario o el servidor. El código del lado del cliente consiste de dos partes:

- La zona pública.
- La zona privada.

4.5 Desarrollo de la zona privada

Como se muestra en la figura 4.5.1 la carpeta *privateZone* que se encuentra en la ruta *public/javascript/privateZone* contiene dentro el archivo *privateZoneFunctions.js* en el cual están definidas todas las funciones y lógica de la aplicación de la zona privada.

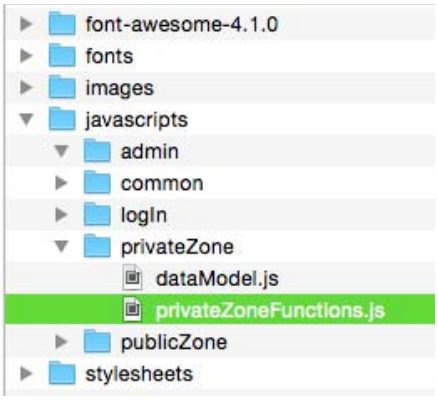


Figura 4.5.1 Localización del archivo donde se encuentran definidas las principales funciones de la zona privada

En el inicio del *script* se encuentran definidas las variables globales del código, a continuación se listan las principales variables con una breve descripción:

```
var mosaicMatrixArray = [];
```

Variable de tipo arreglo donde se almacenan los mosaicos realizados.

```
var selectedMosaicMatrix=0;
```

Variable de tipo numérico donde se indica en que índice del arreglo del mosaicos se encuentra el mosaico seleccionado para ser enviado a la zona pública.

```
var mosaicMatrixBeforeEdit={};
```

Objeto de JavaScript donde se guarda una copia del mosaico antes de ser editado.

```
var initialColor ="white";
```

Variable de tipo cadena donde indica cual será el color default para la elaboración de un mosaico.

```
var OS_Device=getMobileOperatingSystem();
```

Variable de tipo cadena donde se define que tipo de dispositivo está accediendo a la aplicación.

```
var sendCoordinatesEachThisNumber=11;
```

Variable de valor numérico donde se indica la resolución del *trackpad*. Define cuantos disparos del evento movimiento en “x” o “y” en el *trackpad* deben de ocurrir para que se envíe la posición actual del objeto de la zona privada a la zona pública.

```
var socket = io.connect('/');
```

Variable donde se almacena el objeto *io.connect(‘/’)*, el cual permite realizar la comunicación en tiempo real con la zona pública

Modelo de Datos:

```
var mosaicMatrix = {
      mosaic11:{
            triangle1:initialColor,
            triangle2:initialColor,
            triangle3:initialColor,
            triangle4:initialColor,
      },...
```

El modelo de datos es un JSON que contiene objetos de JavaScript con los colores de los triángulos que conforman cada sección del mosaico.

Para la estructura visual de las secciones del mosaico se emplearon SVG (*Scalable Vector Graphics*) el cual es usado para definir gráficas en el formato XML basadas en vectores para la web [25]. La ventaja que presenta el uso de SVG es que no se pierde calidad cuando se realiza un acercamiento o se cambia el tamaño del navegador web.

Dentro de la función $(document).ready(function(){ . . . }) se encuentran definidos los principales eventos y sus manejadores. A continuación se describen las funciones más importantes:

```
function setColorSelected(colorPickerId)
```

Función que guarda en una variable el color elegido por el usuario. Tiene como parámetro el *id* del botón del color que seleccionó el usuario.

```
function changeColorTriangle(id)
```

Función que cambia el color del fondo del triángulo en el mosaico, toma el valor del color seleccionado y se asigna a su correspondiente valor en el modelo de datos. Tiene como parámetro el *id* del triángulo a cambiar el color.

```
function saveMosaicMatrix()
```

Función que guarda en el modelo de datos los colores seleccionados.

```
function sendMosaicMatrix()
```

Función que envía el mosaico a la zona pública

```
function getMobileOperatingSystem()
```

Función que detecta si el usuario accedió a la aplicación mediante un móvil o una computadora.

Este código está escrito en Javascript, y se empleó la librería jQuery para agilizar el desarrollo. El documento *privateZone.ejs* (el cual contiene código HTML) que se encuentra en la carpeta *views* define la estructura visual de la interfaz de usuario. El desarrollo de la zona pública se realizó de manera similar.

4.6 Desarrollo de la zona pública

Dentro del árbol de carpetas del proyecto, en la carpeta *public/javascript/publicJqueryFunctions.js* (figura 4.6.1) se encuentra el archivo donde se definen todas las funciones y lógica de la aplicación de la zona pública.

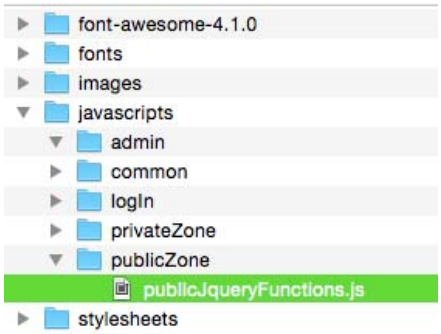


Figura 4.6.1 Localización del archivo donde se encuentran definidas las principales funciones de la zona pública.

En el inicio de este archivo se encuentran definidas las variables globales, a continuación se listan y describen la más importantes:

```
var numberOfColumns=8
```

Variable de tipo numérico donde se define el número de columnas del mural de la zona pública.

```
var numberOfRows = Math.round(numberOfColumns*0.7);
```

Variable de tipo numérico donde se calcula el número de filas del mural de la zona pública.

```
var initialMuralColor = "black";
```

Variable de tipo cadena donde se guarda el color de fondo inicial del mural.

```
var numberOfUsers=20;
```

Variable de tipo numérico donde se define el número de usuarios permitidos por sesión.

```
var socket = io.connect('/');
```

Variable donde se hace referencia al objeto io.connect("/") el cual permite recibir datos de la zona privada

Dentro de la función $(document).ready(function(){ . . . }) se encuentran definidas las funciones que se ejecutarán una vez que el navegador ha terminado de ejecutar todo el código HTML. A continuación una breve descripción de las funciones principales:

```
function setQuadrants(numberOfColumns);
```

Función donde se crean los cuadrantes del mural, los cuales corresponden a un mosaico elaborado en la zona privada. Tiene como parámetros el número de columnas definido en un inicio.

```
function setPublicGrid();
```

Función donde se asigna a cada cuadrante definido un mosaico con el color inicial y con un modelo de datos idéntico al realizado en la zona privada.

```
function onSocketIOMessage();
```

Función que maneja el evento de recibir un mensaje proveniente de la zona privada. Se encarga de transferir los datos recibidos a los elementos HTML para que sean visualizados. Una vez explicadas las variables y funciones que componen la parte del cliente de la aplicación, pasamos a mostrar como fue el resultado del diseño final de la aplicación.

4.7 Diseño final de la primera versión

La mayor parte de los elementos visuales de la zona pública se crearon de manera dinámica mediante código JavaScript, debido principalmente a que se desea la flexibilidad de que el usuario pueda elegir las dimensiones del mural (números de mosaicos que lo componen).

Las siguientes figuras (4.7.1, 4.7.2 y 4.7.3) muestran los diseños finales de la interfaz de usuario para la primera versión de la aplicación tanto de la zona pública como de la zona privada.

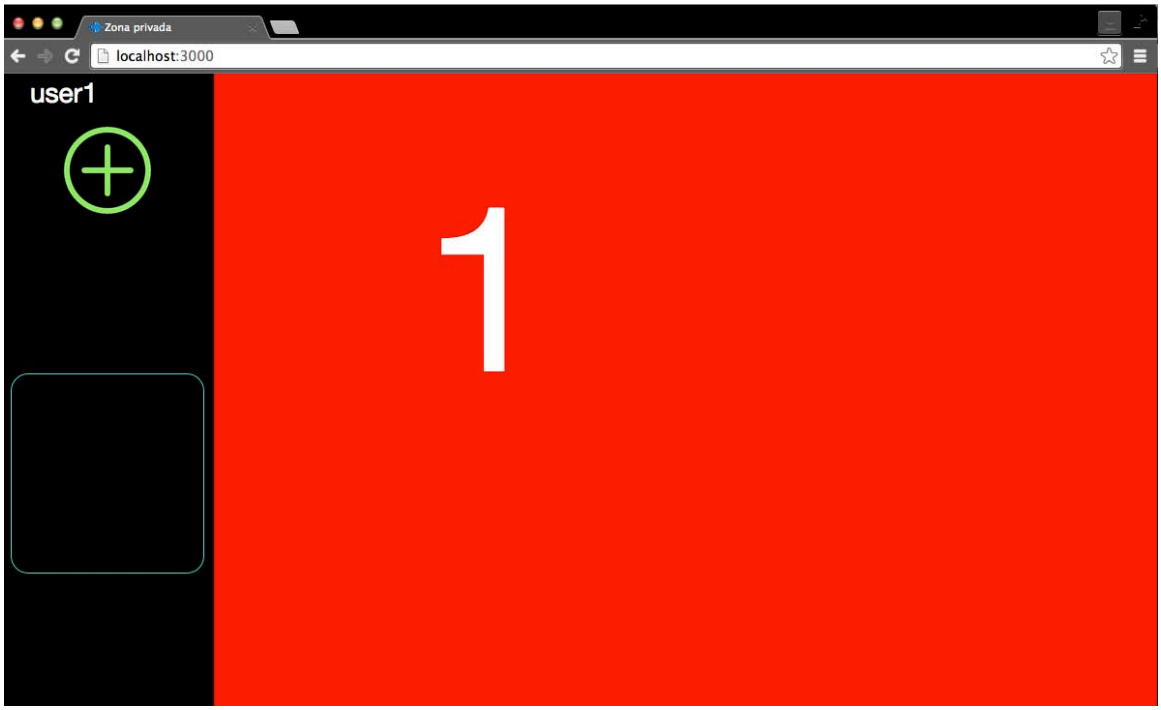


Figura 4.7.1 Interfaz final de *trackpad*.

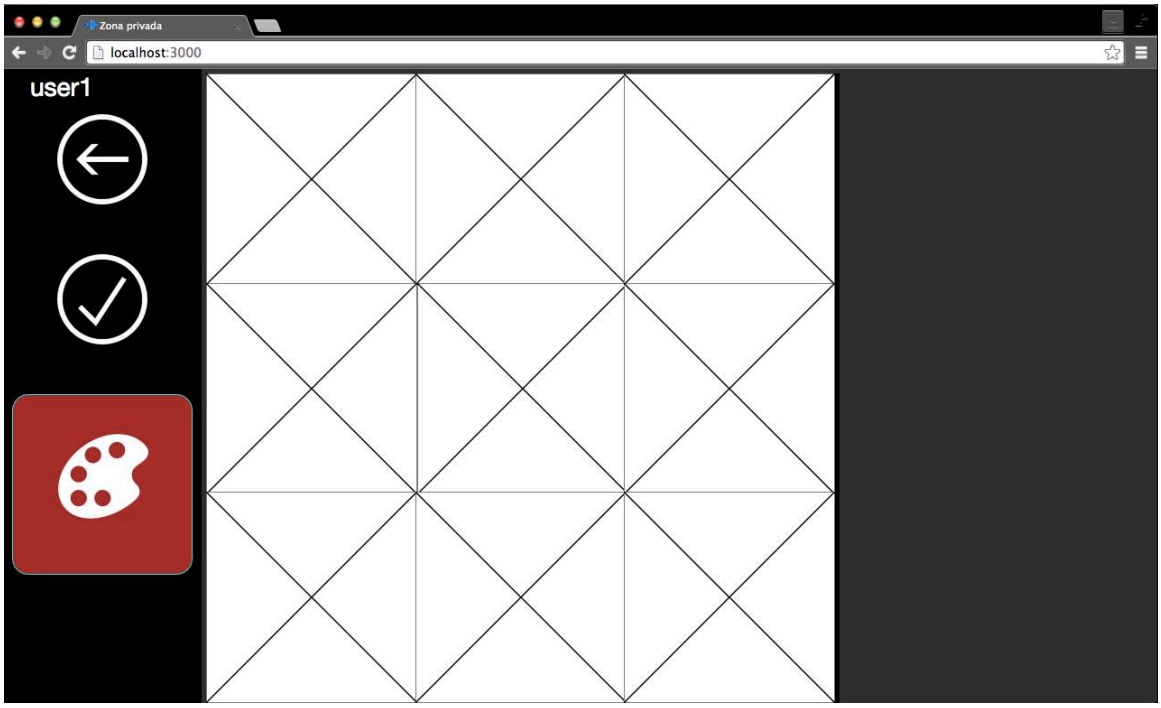


Figura 4.7.2 Interfaz final para la creación de mosaicos

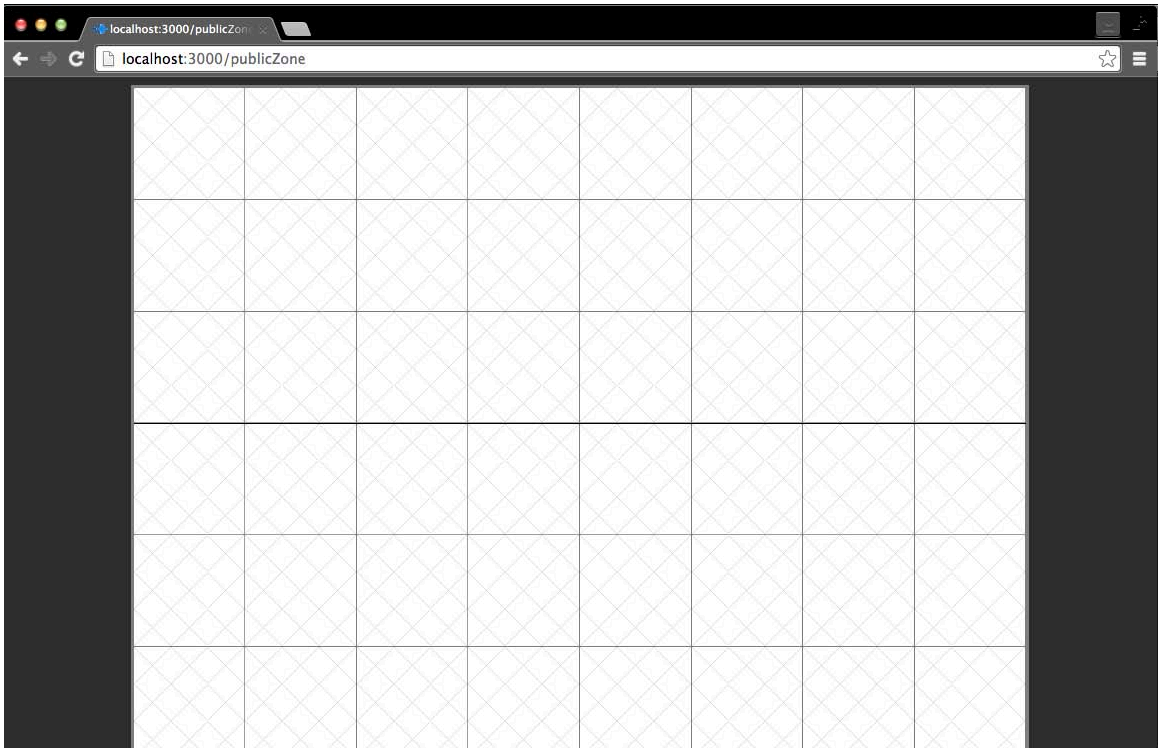


Figura 4.7.3 Interfaz final de la zona pública

Como se puede apreciar en las figuras, los diseños finales fueron muy similares a los realizados en los prototipos. Lo que muestra que el diseño de los prototipos sirvió como guía durante el desarrollo de la aplicación.

Ahora pasaremos al siguiente capítulo donde se explica cuales fueron las herramientas y métodos empleados para desarrollar el código que se encuentra en el servidor, el cual se encarga de sincronizar los datos entre la zona privada y la zona pública.

CAPÍTULO 5

DISEÑO Y DESARROLLO DE BACK-END

El código que se encuentra en el servidor es el encargado de transmitir la información entre la zona pública y la zona privada. Se eligió Node.js como servidor web, Express como marco de trabajo y la librería Socket.io para transmitir datos en tiempo real entre los clientes y el servidor. En este capítulo se explica el funcionamiento básico de las tecnologías seleccionadas, así como la arquitectura y las principales funciones del código, esto con el fin (al igual que el capítulo previo) de que sirva como guía para las personas interesadas en continuar desarrollando la aplicación y como introducción general de las tecnologías empleadas para el lector.

5.1 Estructura del lado del servidor

Iniciaremos describiendo como está distribuido el código del servidor dentro del proyecto. Como se mencionó previamente, mediante el marco de trabajo Express se creó la estructura de carpetas del proyecto. En la carpeta *node_modules* (figura 5.1.1) se encuentran los módulos, estos pueden ser vistos como librerías que se pueden importar para hacer uso de sus funciones. Express y Socket.io son ejemplos de módulos de Node.js, más adelante se explica con más detalle el funcionamiento de los módulos

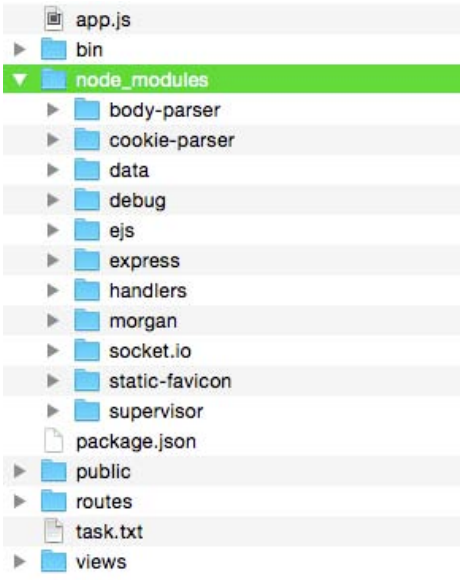


Figura 5.1.1 Módulos de *node.js.*

En la carpeta *routes* (figura 5.1.2) se encuentran las rutas de la aplicación. Las rutas son un elemento clave para el funcionamiento de la aplicación, en estas se encuentran definidas las funciones que serán ejecutadas cuando el usuario acceda a un recurso del servidor. Las

funciones que son llamadas por las rutas se llaman manejadores y se definen en el archivo *indexHandlers.js* que se encuentra en el módulo *node_modules/handlers/*

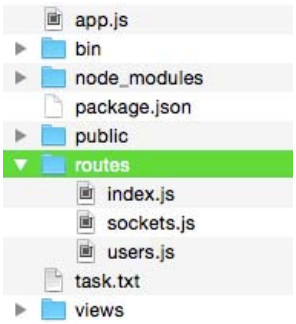


Figura 5.1.2 Carpeta donde se encuentran las rutas a los recursos de la aplicación.

Otro elemento clave es la carpeta *views* (figura 5.1.3), la cual contiene las vistas de la aplicación, por vistas nos referimos a la interfaz que entrega el servidor al cliente. Los archivos con extensión *.ejs (Embedded Javascript)* contienen principalmente código HTML y en menor medida código JavaScript. Al momento de crear la aplicación con Express se define el motor de vistas a emplear. En esta tesis se eligió el motor de vistas *Embedded Javascript*. Un motor de vistas permite crear código HTML de manera dinámica, es decir, se puede hacer uso de variables de JavaScript dentro del código HTML y que estas tomen un valor determinado al momento de ser ejecutadas por el motor de vistas y así esta variable se convierte en un valor de texto en especifico.

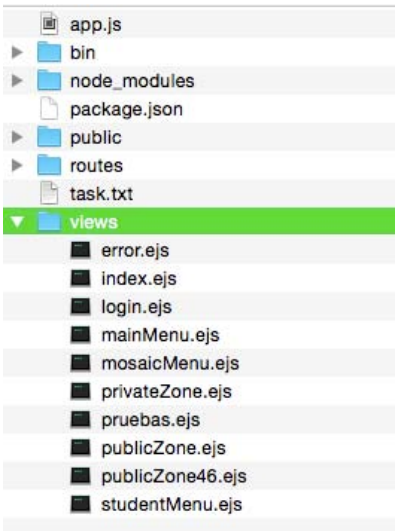


Figura 5.1.3 Carpeta donde se encuentran las vistas.

Una vez declarado donde se encuentran los elementos claves (módulos, rutas, manejadores de eventos y vistas), pasaremos a detallar como se integran estos en la aplicación y cual es su papel dentro del funcionamiento de la misma.

5.2 Módulos de Node.js

Una aplicación desarrollada con Node.js está compuesta principalmente de módulos. Un módulo es una librería de JavaScript que puede ser importada en una aplicación Node.js usando la función *require().* Un módulo puede ser tan simple como una función o tan complejo como un marco de desarrollo web como lo es Express. Existen diferentes maneras de desarrollar módulos, el empleado en esta aplicación se hace mediante el objeto *exports,* al cual se le adjuntan las propiedades y funciones a exportar. Cualquier elemento adjunto al objeto *exports* está disponible como una propiedad o método público, en cambio, una variable definida y no adjuntada al objeto *export* se convertirá en un objeto privado del módulo. A continuación un ejemplo donde se exporta el método *get_name*:

```
1   var usuario = “Bill Gates”;
2   export.get_name = function(){
3           return name;
4   }
```

En la línea 2 del código adjuntamos un nuevo método al objeto *exports*. Una vez definido esto, es posible acceder a este método cuando se importe el módulo dentro de la aplicación, el cual nos regresará el nombre del usuario. Para hacer uso de este fragmento de código bastaría con guardarlo en un archivo con extensión “.js” y colocarlo en la carpeta de *node_modules.*

5.3 Configurando Express

Es indispensable comprender el funcionamiento de Express para comprender en si el funcionamiento de la aplicación. Una gran ventaja de Express es su simplicidad, ya que este se basa únicamente en tres componentes principales:

1) El objeto aplicación (*application*). Este es una instancia de Express, convencionalmente representado por la variable nombrada *app*. Es el objeto principal de la aplicación Express y la mayor parte de la aplicación está construida sobre este objeto

2) El objeto petición (*request*). El objeto petición *http* es creado cuando el cliente hace una petición a la aplicación Express. Este objeto convencionalmente se representa con la variable *req*, que contiene un número de propiedades y métodos relacionados con la petición correspondiente.

3) El objeto respuesta (*response*). Este objeto es creado junto con el objeto petición, convencionalmente se representa con la variable *res* y contiene las propiedades asociadas a una respuesta *http* del servidor.

El archivo "*app.js*" que se encuentra en la raíz del proyecto es el responsable de inicializar la aplicación (figura 5.3.1). Desde este archivo se importan los módulos necesarios mediante el comando *require()* y se definen los parámetros principales de la aplicación .

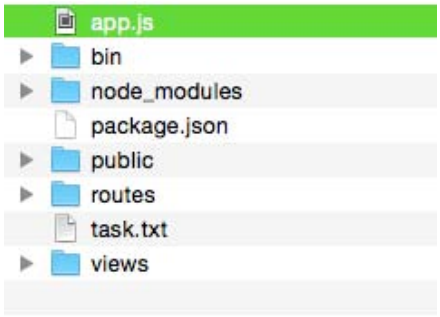


Figura 5.3.1 Localización del archivo principal de la aplicación *app.js*.

A continuación se describen las configuraciones más importantes definidas dentro de este archivo:

```
var express = require('express');
```

Se importa el módulo Express y se guarda el objeto en la variables *express*.

```
var http = require('http')
```

Se importa el módulo *http*, del cual se emplearán las funciones básicas para declarar un servidor *http*.

```
var routes = require('./routes/index');
```

Se importa el módulo de rutas, donde están definidas los manejadores de rutas con sus respectivos URL.

```
var app = express();
```

Se crea el objeto *app*, que tendrá todas las propiedades de Express.

```
app.set('views', path.join(__dirname, 'views'));
```

Se define la localización de las vistas.

```
app.set('view engine', 'ejs');
```

Se define el motor de vistas, en este caso *ejs* (*embedded javascript*)

```
var server = http.createServer(app).listen(3000, function(){
     console.log("Express server listening on port :3000 ")
});
```

Se declara el servidor http en el puerto 3000

```
require('./routes/sockets.js').initialize(server);
```

Se inicia el servidor desde el módulo sockets.js.

Una vez inicializado el servidor, es necesario declarar las rutas para que los clientes puedan acceder a los recursos del servidor.

5.4 Rutas y manejadores

Las rutas son un esquema (basado en URL) que determina como debe de responder el servidor a las peticiones de los clientes. Como se aprecia en la figura 5.4.1 una ruta se definen combinando un método *http* y una función que responda a la llamada de dicho método.

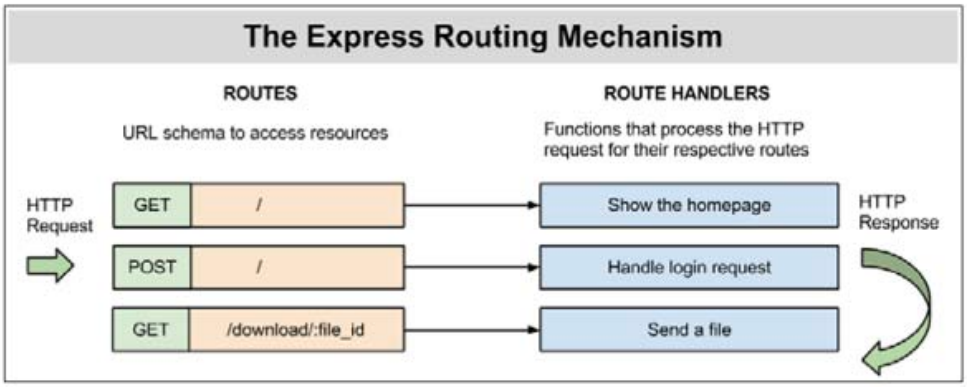


Figura 5.4.1 Mecanismo de rutas en *Express* [22]

Las rutas son definidas usando un método *http* y un patrón de direcciones. Cualquier petición al servidor que corresponda a una ruta definida es dirigida hacia un manejador de rutas asociado. Los identificadores para las rutas pueden ser una cadena o una expresión regular.

Cuando se realiza una petición a un servidor, y este corresponde a una ruta definida, la función *callback* asociada entra en el proceso y manda una respuesta. Esta función *callback* es responsable del comportamiento dinámico de la aplicación. En la figura 5.4.2 se muestra que el objeto ruta puede tener múltiples manejadores, cada uno distinto para cada recurso del servidor que es accedido mediante una petición *http*. En esta aplicación de aprendizaje colaborativo se empleó un modelo sencillo de rutas y manejadores, donde a cada petición *http* le corresponde un manejador que responde ya sea con un JSON o un archivo HTML.

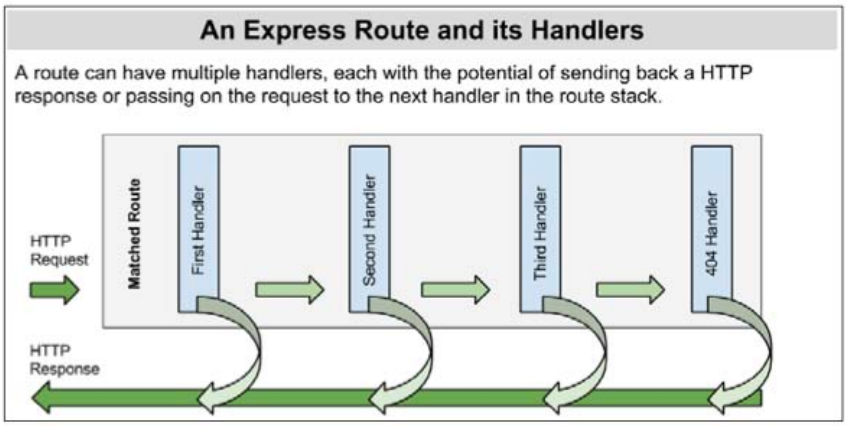


Figura 5.4.2 Objeto ruta y sus manejadores [22]

Una vez listo el servidor para recibir y procesar peticiones http, es necesario implementar canales de comunicación bidireccionales entre los clientes y el servidor para poder sincronizar la información proveniente de la parte pública con la parte privada. Esto se hizo mediante el módulo Soket.io.

5.5 Socket.IO

Como se ha mencionado anteriormente, la aplicación demanda transferencia de datos en tiempo real entre el cliente y el servidor. Normalmente en una aplicación web el cliente es quien hace una petición de datos al servidor y este responde inmediatamente, aunque, como en este caso, también existen aplicaciones en las que se requiere que el servidor envié datos al cliente en cualquier momento. Esto solo es posible mediante un canal bidireccional de comunicación, estos canales son comúnmente llamados *web sockets.*

La librería Socket.io se encarga de esta tarea al servir como canal bidireccional entre un cliente y un servidor. Esta librería debe de estar presente tanto en el lado del cliente como en el lado del servidor. Los eventos emitidos en un *socket* de un lado (cliente o servidor) son atendidos por el correspondiente manejador de eventos del otro lado. Socket.io está construido de tal manera que tanto el cliente como el servidor pueden enviar o recibir mensajes en cualquier momento.

Dentro de esta librería existen principalmente dos tipos de mensajes:

1) El mensaje del sistema: Este mensaje es enviado por el servidor al cliente, indicando cuando un usuario está conectado o desconectado.

2) El mensaje del usuario: Este mensaje es enviado del cliente al servidor y contiene información relevante que el usuario quiere compartir con otros clientes, como la posición del mosaico en el área del *trackpad* y el objeto que contiene la información con los colores del mosaico.

El primer evento que el servidor va a recibir es la de conexión con un nuevo cliente. Este es identificado mediante el evento *connection* del objeto *io.sockets* y notifica a nuestra aplicación que un nuevo cliente ha abierto una nueva conexión y que todo el protocolo de navegación ha sido completado y de ahora en adelante se tiene un canal de comunicación

con este cliente. Mediante el siguiente fragmento de código se declara el manejador del evento *connection* que será disparado:

```
1   io.sockets.on("connection", function(socket){
2   //Agregar código para responder a nueva conexión
3   });
```

En la línea 2 o dentro de los corchetes de la función es donde define que acción realizar cuando se realice una nueva conexión al servidor.

Socket.on es un emisor de eventos que puede disparar diferentes eventos basados en parámetros que le son emitidos. En esta tesis se emplearon principalmente los siguientes eventos:

- `Socket.on('message', function(message, callback) {})`
  El evento mensaje es disparado cuando un mensaje enviado llega desde la función *socket.send*. Los parámetros de la función son el mensaje enviado y la función *callback* la cual es opcional.
- `Socket.on('anything', function(data) {})`
  El evento *anything* puede ser cualquier evento con la excepción de los eventos reservados.
- `Socket.on('disconnect', function() {})`
  El evento desconectado es disparado cuando el cliente o el servidor pierden la conexión con el *socket*.

Una vez definidas las funciones para escuchar eventos, es necesario definir ahora los métodos para enviar mensajes. A continuación se describen mediante fragmentos de código los métodos empleados para transmitir mensajes en esta tesis:

1). Enviar mensajes del lado del servidor

```
1     socket.send(JSON.stringify({
2          type:'serverMessage',
3          message: 'GOYA GOYA UNIVERSIDAD!!'
4     }));
```

El método *socket.send* en la línea 1 enviará el mensaje sobre el *socket*, el cual disparará el evento *message* en el cliente. El mensaje enviado debe de ser de tipo cadena, por ello se implementa la función *JSON.stringify*.

2) Recibir y enviar mensajes del lado del servidor

```
1    socket.on('message', function(message){
2        message= JSON.parse(message);
3        socket.broadcast.send(JSON.stringify(message));
4    });
```

Este fragmento de código se encuentra en el lado del servidor. En la línea 1 la función *socket.on* recibe dos parámetros, el nombre del evento (‘*message*’) y el manejador de evento para ello (una función). Como el mensaje del cliente llega como tipo cadena se necesita convertir el mensaje a un JSON (línea 2). El siguiente paso es enviar este mensaje a todos los usuarios conectados (clientes). Para esto Socket.io provee el objeto *broadcast*, declarado en la línea 3, el cual se encarga de enviar el mensaje a todos los clientes conectados. Así los datos enviados por la zona privada son recibidos por el servidor y reenviados a todos los clientes, pero solo la zona pública (que es también un cliente) tiene un evento asignado para escuchar este tipo de mensaje (el enviado desde la zona privada).
El código donde se implementaron las funciones de la librería Socket.io se encuentra en la carpeta de rutas en el archivo *sockets.js* (figura 5.5.1) y es donde se declara la conexión por *sockets* entre los clientes y el servidor.

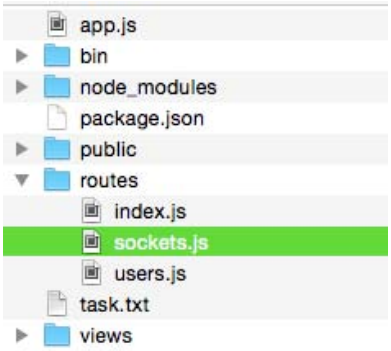


Figura 5.5.1 Localización donde se encuentra la implementación de los *sockets*

A continuación se muestra el código completo del archivo socket.io donde se encuentra definida la lógica que se encarga de la transferencia de datos entre los clientes y los servidores:

```
1.  // Sockets File
2.    var io = require('socket.io'); // importar módulo socket.io
3.    exports.initialize = function(server) { //
4.       io = io.listen(server);
5.       io.sockets.on("connection", function(socket){
6.           socket.send(JSON.stringify(
7.           {type:'serverMessage', message: 'Starting App' }
8.           ));
9.           socket.on('message', function(message){
10.             console.log("message received")
11.             message= JSON.parse(message);
12.             if(message.type == "userMessage"){
13.                   socket.broadcast.send(JSON.stringify(message));
14.             }// close if
15.          });
16.     }); // close io.sockets.on
17.   };
```

En la línea 2 se crea el objeto *"io"* que contará con todos atributos y métodos del módulo *socket.io;* en la línea 4 se inicializa el socket en el lado del servidor para que pueda recibir mensajes del cliente; en la línea 5 se define la función para el evento de conexión de un nuevo usuario y se define también la función correspondiente a ejecutar cuando se dispare dicho evento; en la línea 10 se declara el código que atiende al evento "mensaje" y su respectiva función a llamar cuando se dispare dicho evento; en la línea 14 se envía un mensaje en formato JSON a todos los canales de comunicación que estén abiertos, estos pueden ser clientes u otros servidores, en nuestra aplicación nos interesa que sea para los clientes.

En resumen, se presentó una introducción básica al funcionamiento de Node.js (servidor web), Express (marco de trabajo de node.js) y Socket.io (encargado de la transferencia de datos) así como su implementación en esta tesis.

Una vez desarrollado tanto el *front-end* y el *back-end* de la aplicación, se tiene la primera versión terminada. El siguiente paso es hacer pruebas con y sin usuarios y en diferentes procesadores o sistemas embebidos para medir y analizar el comportamiento de la aplicación para finalmente obtener resultados del desempeño de la misma como herramienta para el aprendizaje colaborativo.

CAPÍTULO 6

PRUEBAS CON USUARIOS Y ANÁLISIS DE RESULTADOS

Para evaluar la propuesta desarrollada en este trabajo, se llevó a cabo un Caso de Estudio con alumnos y profesores de nivel primaria. En efecto, durante los meses de febrero a junio (2015), se estuvo acompañando a profesores de quinto y sexto año de primaria, para desarrollar secuencias didácticas que involucraran el uso de la aplicación colaborativa. Durante este transcurso se añadieron distintas funcionalidades a la aplicación (como compartir texto e imágenes) con el fin de que se adecuara a un rango más amplio de actividades colaborativas.

Las primeras pruebas que se realizaron a la aplicación fueron sin usuarios y estaban enfocadas a verificar que la aplicación cumpliera satisfactoriamente con los requisitos establecidos durante el diseño. Una vez que se corroboró el funcionamiento correcto de la aplicación se pasó a realizar pruebas con usuarios. La primera prueba con usuarios se realizó en el evento "FORO CONMEMORATIVO 2020: Una visión de Futuro de la UNAM", organizado por Fundación UNAM. Esta fase tuvo como propósito probar la usabilidad de la aplicación con los usuarios y tener noción acerca de que tan intuitiva es la interfaz de usuario. Finalmente, se tuvo la oportunidad de participar en el "Programa Piloto de Inclusión Digital", dirigido por la Coordinación de Estrategia Digital Nacional de la Oficina de la Presidencia de la República, lo que nos permitió trabajar con profesores y alumnos de una escuela primaria, integrando la propuesta a las actividades dentro del aula. Estas tres fases se detallan en las siguientes secciones.

6.1 Pruebas del funcionamiento de la aplicación sin usuarios

Antes de realizar pruebas con usuarios fue necesario comprobar el funcionamiento correcto de la aplicación. El objetivo de estas primeras pruebas fue principalmente corroborar que existiera una transferencia de datos fluida entre la zona privada y la zona pública y que esta información (mosaicos enviados y posición del cursor del usuario) se visualizara correctamente. La principal adversidad que se planteó desde un inicio fue la intensidad de tráfico que provoca la transferencia de datos en tiempo real en una red, además del poder de procesamiento que demanda esta transferencia de datos tanto al servidor como a la zona pública. Esto nos dirigió a realizar pruebas con diferentes dispositivos (raspberry Pi, UDOO, BeagleBone Black) y diferentes tipos de redes (redes de área local e Internet).

Una primera prueba relevante fue la de comprobar cuantos usuarios simultáneos era capaz de soportar la aplicación, y es que la aplicación se encuentra limitada de manera muy importante por el hardware (procesador del dispositivo que ejecuta la aplicación, y la calidad de la red bajo la que estén conectados todos los participantes). Si bien este primer elemento puede quedar fuera del control de nuestra aplicación, existe un segundo factor que sí es posible controlar y cuyo impacto en el desempeño final es un tanto menor. Nos referimos al número peticiones que hace el sistema por unidad de tiempo para cada uno de los usuarios. Entre más se optimice (reduzca) el número de peticiones http que se realiza por usuario, mayor podrá ser el número de usuarios que soporte la aplicación.

Estas pruebas sin usuario se realizaron inicialmente en una máquina *MacBook Pro* modelo 2010 con procesador 2.4 GHz *Core 2 duo*, 4 GB de RAM. También se empleo una red local soportada por un *access point* marca Linksys modelo WRT54GL. Inicialmente se encontró que con uno a tres usuarios interactuando al mismo tiempo el funcionamiento de la aplicación era fluido, pero al ingresar un usuario más, decaía la velocidad con la que la zona pública reflejaba las interacciones de los usuarios. Esto se debía a que el total de número de peticiones http que realizaban los usuarios superaba la capacidad de respuesta del servidor lo que causaba que la aplicación se alentara. El problema se arregló disminuyendo el número de peticiones por usuario. Inicialmente el sistema estaba configurado para que por cada movimiento en un pixel que realizaba un usuario en la zona privada se realizara una petición http para enviar las nuevas coordenadas. Se redujo el número de peticiones http haciendo más eficiente el algoritmo de actualización del cursor.

Una vez mejorado el software, se procedió a realizar pruebas con diferentes sistemas embebidos, a continuación se muestra una tabla con los resultados obtenidos:

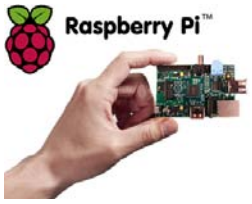
Sistema Embebido	**Procesador**	**Número de usuarios que soporta**	**Imagen**
Raspberry pi Model b+	900 MHz quad-core ARM Cortex-A7	2-3	
Beaglebone Black	AM335x 1GHz ARM® Cortex-A8	2-3	
UDOO	Freescale i.MX 6 ARM Cortex-A9 Quad core 1GHz	3-4	

Tabla 6.1.1 Sistemas embebidos donde se realizaron pruebas

De la tabla 6.1.1 se puede concluir que por el momento no es viable usar alguno de los sistema embebido en donde se realizaron pruebas, ya que un requisito de la aplicación es que el hardware permita que entre uno a seis usuarios puedan interactuar al mismo tiempo. El máximo número de usuarios que se alcanzó fue con el UDOO (el que tiene mejor procesador) y fue de cuatro usuarios.

6.2 Pruebas con el usuario final

Las primeras pruebas que se realizaron con usuarios fueron en el evento “FORO CONMEMORATIVO 2020: Una visión de Futuro de la UNAM”. El objetivo de estas pruebas fue percibir que tan intuitiva y fácil de usar era la interfaz de usuario de la aplicación. Se les pidió a los asistentes del evento (adultos en su mayoría) que crearan una sección del mural (un mosaico) en su celular (que funcionó como zona privada) y que lo compartieran a la zona pública (una pantalla). Después de una breve explicación del funcionamiento de la interfaz, encontramos que los usuarios no tuvieron dificultades para

usar la interfaz al crear los mosaicos. Fue claro percatarse de aquellas personas que estaban familiarizadas con el uso de tabletas y teléfonos inteligentes. Estas incluso no necesitaron de una explicación del funcionamiento de la interfaz e hicieron uso de la aplicación mediante su intuición. En la figura 6.2.1 se muestra el mural colaborativo final realizado por los usuarios.

Figura 6.2.1 Zona pública de las primeras pruebas realizadas con usuarios

Una vez que se considero que la interfaz de usuario era suficientemente agradable e intuitiva, se trabajó con profesores y alumnos de la escuela primaria *Christel House*, diseñando actividades colaborativas que implicaran el uso del mural colaborativo. Realizar estas pruebas fue posible debido a la participación que se tuvo en el "Programa Piloto de Inclusión Digital", dirigido por la Coordinación de Estrategia Digital Nacional de la Oficina de la Presidencia de la República.

Al inicio de las pruebas en la escuela primaria con niños de quinto y sexto grado se les explicó a los alumnos en que consiste y como se utiliza la aplicación, después se les pidió que completaran el mural colaborativo desplegado en la zona pública. Durante el transcurso de la actividad ocurrieron discusiones, negociaciones y posteriormente un reparto de tareas a elaborar. Se observó que el interfaz les resultó fácil de usar ya que bastó una breve explicación del funcionamiento de la aplicación para aprender a usarla.

Figura 6.2.2 Alumnos de quinto de primaria usando la aplicación.

En la figura 6.2.2 se puede observar a un grupo de cuatro alumnos de quinto de primaria usando la aplicación, cada alumno cuenta con una tableta (con sistema operativo *Android*) como su zona privada, y en el pizarrón se proyecta la zona pública, donde se muestra el mural a replicar en equipo.

6.3 Modificaciones a la aplicación

Posterior a las primeras pruebas realizadas en el aula, se modificó la aplicación para ahora poder compartir texto e imágenes desde la zona privada a la zona pública, esto con el fin de ampliar las actividades colaborativas que se pueden realizar. En la zona privada (figura 6.3.1) se agregó una interfaz donde el usuario puede escribir texto para compartirlo a la zona pública.

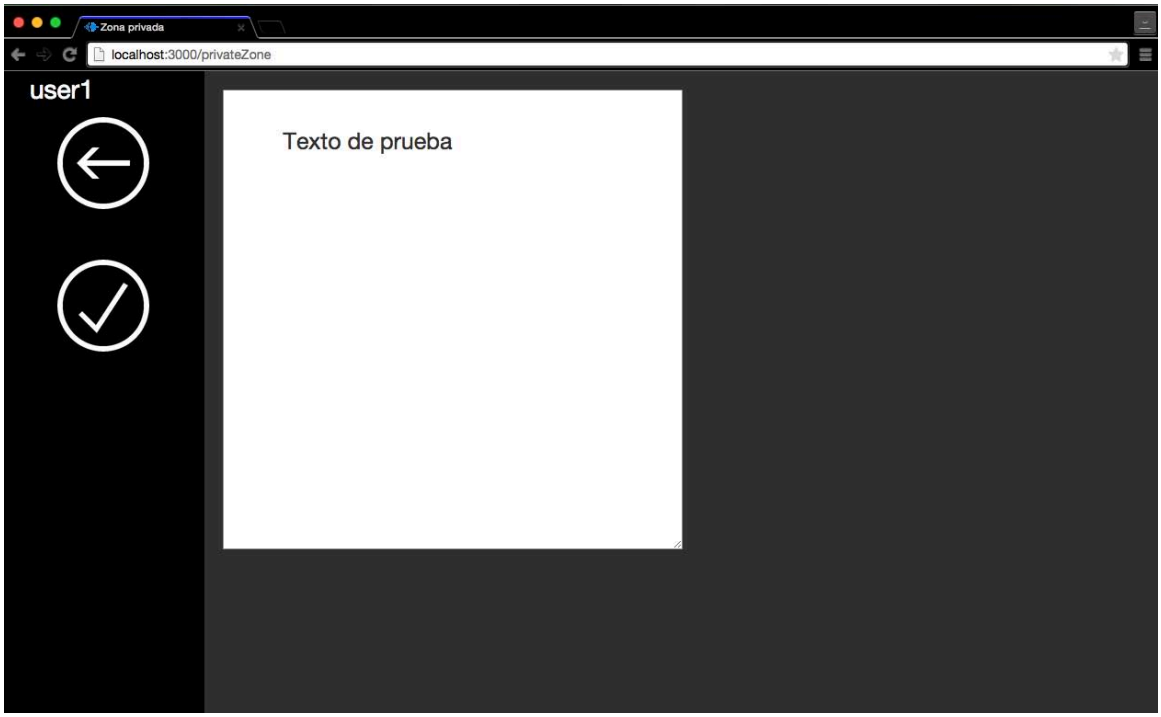


Figura 6.3.1 Interfaz de la zona privada donde el usuario puede crear texto

De igual manera, como se muestra en la figura 6.3.2 la zona pública se modificó para ahora poder visualizar el texto compartido por los usuarios.

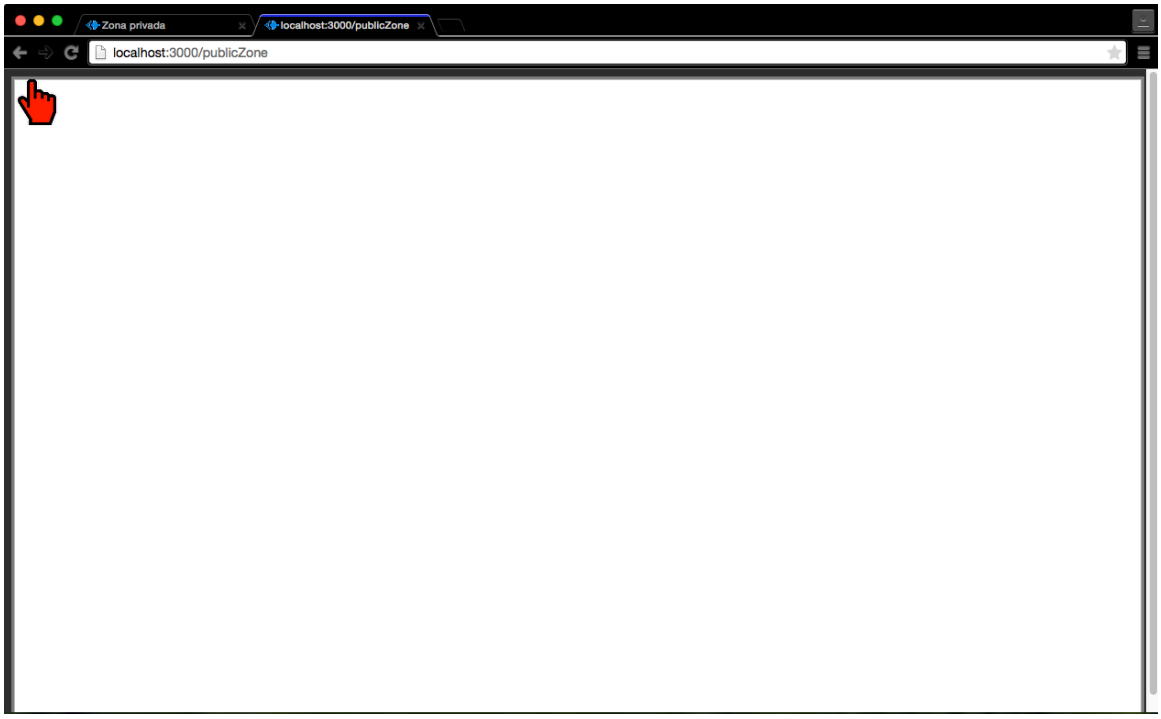


Figura 6.3.2 Zona pública compuesta por un mural en blanco.

Estas nuevas funcionalidades, compartir texto e imágenes, vuelven más versátil la herramienta, ya que ahora es posible usar la aplicación para un rango más amplio de actividades colaborativas en el aula. En la escuela primaria *Christel House* se utilizó para tres distintas actividades que involucraron los siguientes temas: calidad de vida, contaminación ambiental y las partes de un cuento.

En la primera actividad se les pidió que en equipo discutieran cuales son los aspectos más importantes relacionados con la calidad de vida. Una vez discutido el tema y escritas sus ideas y opiniones, estas fueron compartidas y expuestas con el resto de la clase a través de la aplicación. El momento de la discusión se muestra en la figura 6.3.3.

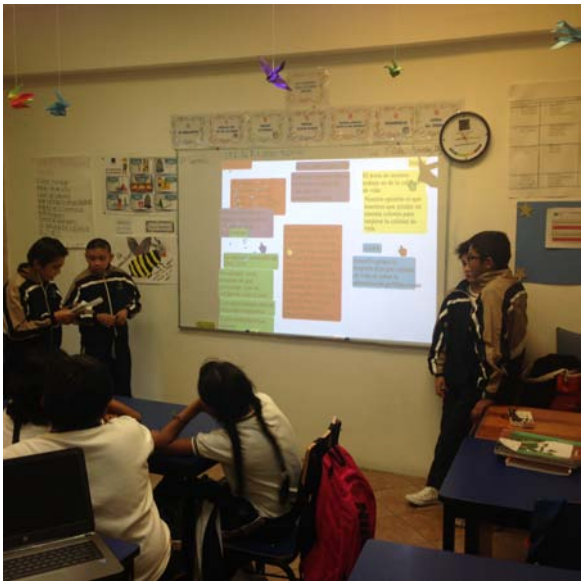


Figura 6.3.3 Alumnos discutiendo el tema de calidad de vida mediante el mural colaborativo.

En la segunda actividad se les pidió a los alumnos que realizaran un cuento en equipo, los alumnos discutieron, negociaron y al final se asignaron las partes del cuento que desarrollaría cada integrante (figura 6.3.4).

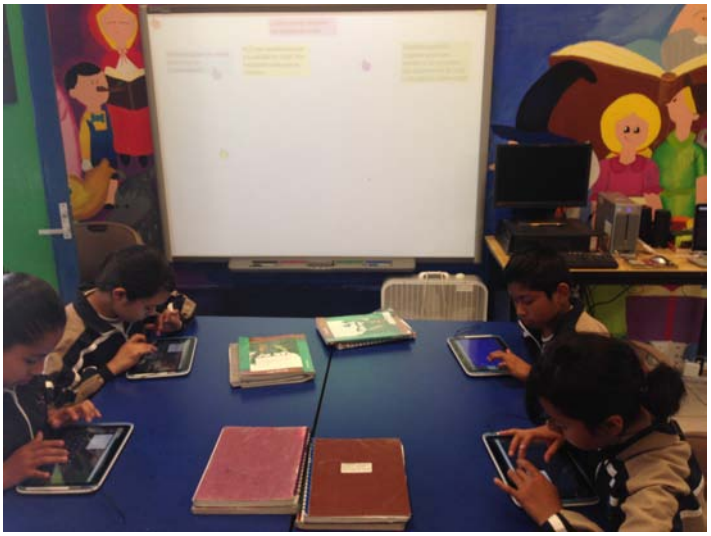


Figura 6.3.4 Alumnos escribiendo un cuento en colaboración

A continuación en la figura 6.3.5 se muestra el cuento realizado por los alumnos, cada color corresponde a un usuario.

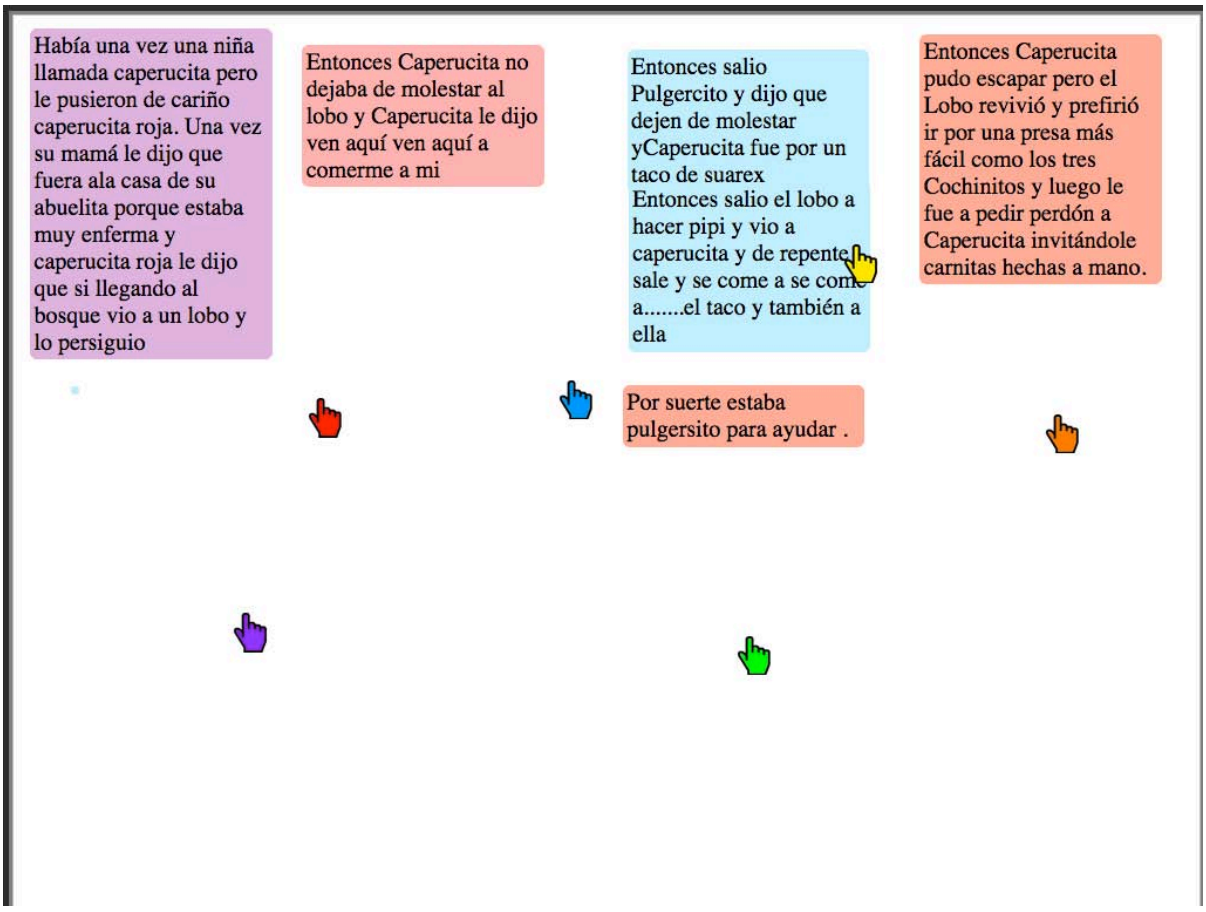


Figura 6.3.5 Zona pública donde se muestra el cuento escrito colaborativamente por alumnos de quinto de primaria

En la tercera actividad se les dio instrucciones a los alumnos para que en equipo hicieran una lluvia de ideas acerca de cómo podrían disminuir los problemas ambientales. Cada equipo eligió un tipo de contaminación distinto (contaminación del aire, basura, contaminación del agua, deforestación, cambio climático). En la figura 6.3.6 se muestra el mural donde los alumnos compartieron sus soluciones.

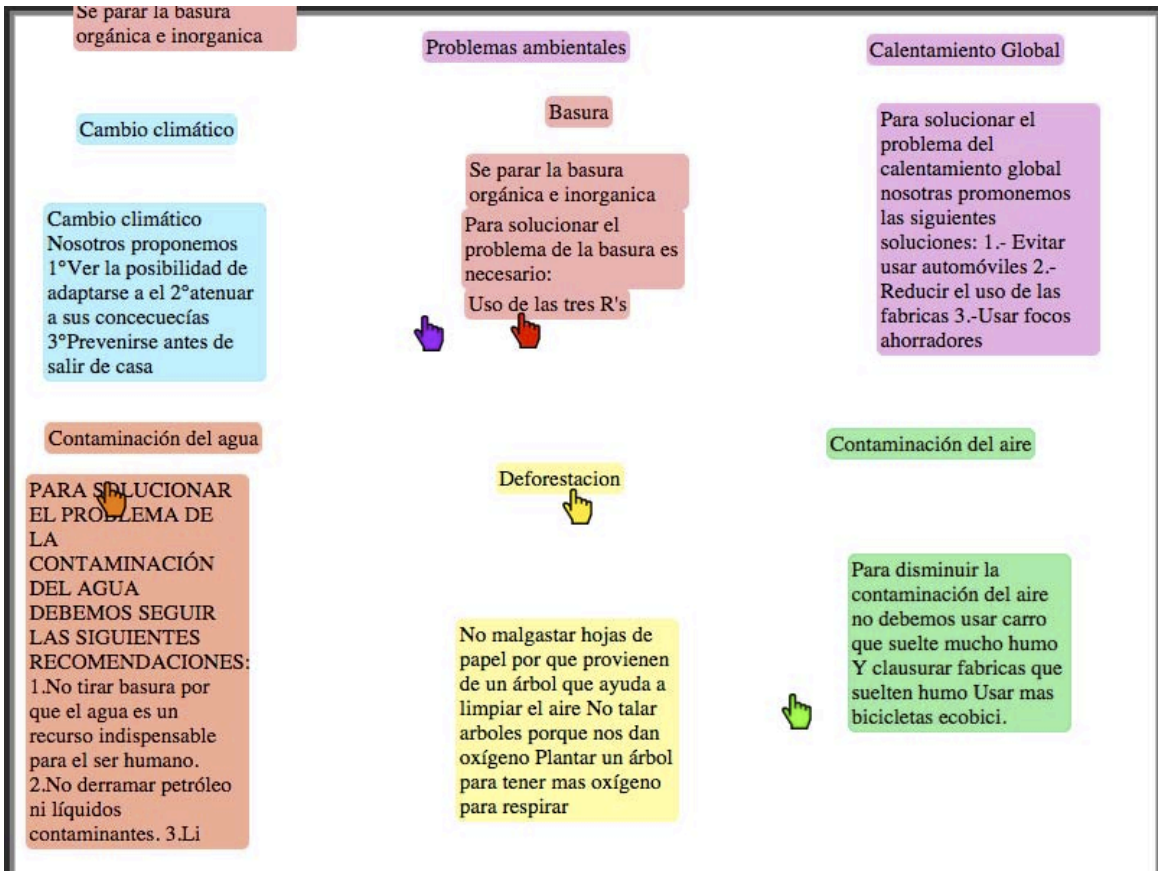


Figura 6.3.6 Zona pública donde se muestra las soluciones para combatir los problemas ambientales

Una vez realizadas las pruebas de la aplicación como herramienta para el aprendizaje colaborativo, pasamos a analizar los resultados que obtuvimos de las mismas.

6.4 Análisis de resultados

En el capítulo 2 se mencionó que los cuatro criterios a considerar en un escenario educativo para aumentar la probabilidad de que exista un aprendizaje colaborativo son:

1. El conjunto de elementos que crean una **situación** apropiada para el aprendizaje colaborativo (similitud entre los integrantes del grupo, grado de repartición de tareas).

2. Las **interacciones** que provocan mecanismos de aprendizaje (negociación, sincronización, interactividad).
3. Los **procesos** de aprendizaje (modelado mutuo de un problema, ampliación de conceptos).
4. Los **efectos** del aprendizaje colaborativo (mejora de habilidades de trabajo en equipo, comprensión más amplia de un concepto).

En especifico estas interacciones son:

- **Interactividad**: Indica el grado de interacción entre los pares, este no está definido por la frecuencia de sus interacciones, si no más bien por como estas interacciones influencian los procesos cognitivos de los integrantes del grupo.
- **Sincronización**: Realizar una actividad en equipo implica comunicación sincronizada, se puede considerar como una regla de comunicación en la que un integrante del grupo emite un mensaje y este espera a que su par lo reciba y lo procese tan rápido como le sea llegado. Dicho de otra manera, se busca un razonamiento sincronizado por parte de los integrantes.
- **Negociación**: Se espera que en una situación colaborativa los integrantes del equipo puedan argumentar su punto de vista, justificarlo, negociar e intentar convencer al otro integrante.

Comenzando por el criterio que crea una situación apropiada para el aprendizaje colaborativo encontramos que estuvieron presentes los siguientes elementos durante las pruebas realizadas con los usuarios finales:

- Simetría de acción. Todos los integrantes del grupo podían realizar las mismas acciones, ya que cada integrante contaba con su propia zona privada para trabajar y compartir información.
- Simetría de conocimientos. Los integrantes del grupo tenían en general el mismo nivel de conocimientos ya que todos se encontraban en el mismo nivel de primaria.
- Simetría de estatus. En este criterio existió una ligera asimetría, ya que al elegir niños al azar, algunos contaban con diferentes estatus dentro del grupo, por ejemplo, algunos niños desempeñaban tareas de lideres durante la realización de las tareas, lo que

aumentó su participación en la actividad, mientras otros niños de perfil más tímido participaban menos.

Estas simetrías afectaron directamente la manera en que interactuaron lo usuarios. Al contar con los mismos conocimientos y libertad de acción, existió una auto organización por parte de los niños, lo que derivó en el siguiente criterio del aprendizaje colaborativo: interacciones colaborativas. En este criterio es importante aclarar que sus elementos (interactividad, sincronización, negociación) variaron en función de la actividad y temas a desarrollar (calidad de vida, contaminación ambiental y las partes de un cuento), a continuación se explican con mayor detalle:

- **Interactividad**: Si bien hubo interactividad en los tres temas que se desarrollaron mediante el mural colaborativo, existió un mayor grado de interactividad en la actividad de desarrollar un cuento.
- **Sincronización**: Existió una sincronización casi al mismo nivel en las tres actividades, aunque el desarrollo del cuento demandó una sincronización de ideas más elaborada.
- **Negociación**: En este aspecto, el desarrollo del cuento fue la actividad con menor negociación, ya que cada integrante era libre de generar su propia parte del cuento, solo había negociación al momento de tener que conectar cada parte del cuento. En cambio en las otras dos actividades (discusiones acerca de cómo mejorar la calidad de vida y cómo disminuir la contaminación ambiental) hubo una mayor negociación ya que demandaban repartición y discusión de opiniones para dar con soluciones.

Los últimos dos criterios del aprendizaje colaborativo corresponden a los mecanismos de aprendizaje que pueden estar presentes en las interacciones colaborativas y a los efectos del aprendizaje colaborativo. Estos dos criterios no se expresan con la misma claridad que los dos criterios anteriores. Podríamos inferir y suponer (debido a que no sabemos con certeza que procesos cognitivos ocurrieron en la mente de los niños) que estuvieron presentes los siguientes mecanismos de aprendizaje derivados de interacciones colaborativas, a continuación una posible justificación:

- **Inducción**: Los niños podrían haber adquirido una representación más amplia de un concepto al construir una nueva definición que integrara las definiciones de los demás integrantes del grupo.
- **Carga cognitiva**: En colaboración, el reparto de actividades redujo la cantidad de información a asimilar por el niño, lo que pudo haber reducido la carga cognitiva y hacer más fácil el aprendizaje.
- **Explicación**: Al haber una auto organización por parte de los niños, en la mayoría de las actividades (partes de un cuento y contaminación ambiental) nunca se les pidió explícitamente que dieran su explicación al grupo. Sin embargo, en la actividad donde si era obligatorio dar una explicación (argumentar cómo podemos mejorar nuestra calidad de vida) hubo explicaciones y argumentaciones muy pobres, por lo que podríamos suponer que este mecanismo de aprendizaje no fue muy enriquecedor en los procesos cognitivos de los niños.
- **Conflicto:** De manera similar que el mecanismo anterior, generalmente hubo poca discusión entre las ideas de los integrantes del equipo (en parte por la naturaleza de las actividades desarrolladas).

Aquí es donde se vuelve más evidente que las interacciones colaborativas y los procesos de aprendizaje involucrados dependen en gran medida de la actividad a desarrollar. Basándonos en las tres actividades anteriores pudimos apreciar que tanto las interacciones y los posibles mecanismos de aprendizaje varían dependiendo de la naturaleza de la actividad. Por ejemplo, en la actividad de crear un cuento, hubo poca interactividad y negociación, ya que los niños tenían la libertad de echar a andar su imaginación y crear su propia parte del cuento. Pero hubo una fuerte sincronización de ideas, ya que debía de haber congruencia al momento de unir las partes del cuento. Otro ejemplo se da en la actividad de calidad de vida, donde era obligatorio dar una justificación de la opinión, es más probable que aquí se hayan dado los mecanismos de aprendizaje de explicación y conflicto.

Por último, pasando al cuarto criterio (efectos del aprendizaje colaborativo) este se divide en dos tipos de evaluaciones. La primera es evaluar los conocimientos adquiridos por los integrantes del grupo mediante exámenes realizados después de la actividad. La segunda consiste en la medición o apreciación de una mejora en el desempeño del grupo, dentro de

esta evaluación se debe de verificar si el desempeño del grupo ha aumentado o si los miembros del grupo han obtenido o mejorado sus habilidades para colaborar.

En este trabajo de tesis optamos por la segunda evaluación de manera cualitativa, observando que existe una mejora evidente en las habilidades de los usuarios para trabajar en equipo. Esto basado en la observación en que mientras transcurrían las actividades era más natural y fluida la manera en que los niños tomaban decisiones, repartían actividades, interactuaban y negociaban.

En el siguiente capítulo se sintetizan las conclusiones obtenidas de estos análisis y del desarrollo de la aplicación en general.

CAPÍTULO 7

CONCLUSIONES

Los tres principales temas que aborda esta tesis son el desarrollo de Software, pruebas con Hardware y el uso de la aplicación como herramienta para el aprendizaje colaborativo. De cada tema se puede concluir lo siguiente.

Conclusiones del desarrollo del Software

Al comenzar con el diseño de la aplicación, fue de gran ayuda el uso de la herramienta *moqups* para realizar el prototipo del interfaz de usuario. Este prototipo sirvió como guía para las modificaciones que se realizaron al interfaz durante el desarrollo y pruebas. En la etapa del desarrollo de la aplicación, las tecnologías que se emplearon para el *front-end* fueron HTML5, CSS3 y Javascript mediante la librería de jQuery. Estas tecnologías, al ser el estándar más empleado para el desarrollo web, nos brindaron la ventaja de contar con una amplia documentación. Por otra parte, si bien el uso de la librería jQuery permitió en un inicio un rápido desarrollo, a medida que creció el proyecto se volvió difícil de mantener y agregar nuevas funcionalidades. Esto debido a que jQuery no se apega a un estándar de desarrollo, cada usuario de la librería puede hacer uso libre de sus funciones. Un reciente marco de trabajo (creado en 2009) para el *front-end* llamado Angular.js (desarrollado por Google) está ganando popularidad en aplicaciones web. El marco de trabajo Angular.js se apega al estándar de desarrollo MVC (Modelo-Vista-Controlador) lo que lo vuelve más fiable y facilita el mantenimiento y modificaciones a las aplicaciones web. El código de esta tesis se está migrando a Angular.js.

Para el desarrollo del *back-end* se emplearon Node.js como servidor web, Express como marco de trabajo y Socket.io para la transferencia de datos entre la zona pública y la zona privada. Node.js junto con Express resultaron ser una solución muy acorde a las necesidades del proyecto (concurrencia de múltiples usuarios y transferencia de datos en tiempo real). Además Node.js tiene la gran ventaja de ser tan ligero que es posible instalarlo en sistemas embebidos. La desventaja que tiene Node.js es que no cuenta aún con un IDE (*Integrated Development Environment*) robusto como lo tienen otros servidores web que usan Java, C# o PHP como lenguaje de programación (Apache, Glassfish, IIS). El uso de un IDE facilita en gran medida el desarrollo de aplicaciones y ayuda a depurar y dar mantenimiento al código al momento de desarrollar. Por su lado Socket.io se encargó exitosamente de crear un canal bidireccional entre el servidor y los clientes.

Conclusiones de pruebas con distintos dispositivos

Uno de los objetivos al buscar mejorar la aplicación Escritorio Colaborativo Aumentado (ECA) por medio del desarrollo del Mural Colaborativo fue aumentar la portabilidad de la aplicación. Se busca que la aplicación se ejecute en un dispositivo que quepa en la palma de la mano y se pueda llevar e implementar en cualquier salón de clases. Es por esto que se realizaron pruebas con distintos sistemas embebidos. Los sistemas embebidos que se probaron fueron: Raspberry PI, Beaglebone black y UDOO. Después de hacer uso de la aplicación en estos tres dispositivos, encontramos que la aplicación se mantiene estable hasta con tres usuarios (cuatro para el caso del UDOO). Por otro lado, al hacer pruebas en procesadores de computadoras de gama media (*core 2 duo, i3, i5* de Intel) la aplicación mostró un funcionamiento estable y la transferencia de datos entre la zona pública y la zona privada fue fluida hasta con seis usuarios a la vez.

Otras pruebas importantes relacionadas con el hardware fueron sobre la red en la que funcionaba la aplicación. Se realizaron pruebas con una red de área local (LAN) y a través de Internet. Para la red local se utilizó un *access point* marca Linksys modelo WRT54GL, el cual mostró un buen desempeño en la transferencia de datos entre la zona pública y la zona privada. Para las pruebas realizadas a través de internet se instaló la aplicación en un servidor de la UNAM con IP pública. Estas pruebas mostraron distintos resultados. Encontramos que el desempeño de la aplicación se ve afectado por la latencia de la transferencia de datos. Cuando los clientes se encuentran geográficamente cerca del servidor, la transferencia de datos es fluida, pero al alejarse los clientes del servidor, la comunicación se ve afectada por la latencia, haciendo que la aplicación tarde en reflejar la información que los clientes envían desde la zona privada a la zona pública.

Conclusiones de las pruebas realizadas con los usuarios

Las diferentes evaluaciones permitieron, en primer lugar, asegurarnos de que la herramienta tenía un funcionamiento fluido y agradable para los usuarios. En la segunda fase, pudimos observar que la herramienta es asequible para los usuarios, que la propuesta de interacción entre la zona privada y la zona pública es fácilmente comprensible, y que no genera incertidumbre o desasosiego en los usuarios. Finalmente, en la tercera fase,

pudimos obtener evidencia de la utilidad de la herramienta como una plataforma de apoyo al trabajo colaborativo en el salón de clases, permitiendo a los profesores adaptarla a sus necesidades específicas, y a los niños interactuar, discutir y negociar, sin tener que ocuparse de cómo funciona la tecnología.

De las pruebas realizadas con distintas actividades didácticas observamos que el emplear la aplicación como herramienta para el aprendizaje colaborativo da resultados positivos, ya que durante el transcurso del uso de la aplicación los integrantes del equipo mostraron una mejora en las habilidades para trabajar en equipo como lo son la negociación, la discusión, la sincronización de ideas, la explicación y la distribución de tareas. También se comprobó que los criterios que definen a el aprendizaje colaborativo (la situación, las interacciones, los mecanismos de aprendizaje y el efecto del aprendizaje colaborativo) dependen en gran medida de la actividad a desarrollar con la aplicación Mural Colaborativo. Por lo tanto podemos concluir, que a través de las actividades adecuadas es posible lograr un aprendizaje colaborativo mediante la aplicación desarrollada en esta tesis.

BIBLIOGRAFÍA

[1] Dillenbourg, P. (1999). What do you mean by collaborative learning?. Collaborative-learning: Cognitive and Computational Approaches., 1-19.

[2] Johnson, D. W., Johnson, R. T., & Holubec, E. J. (1999). El aprendizaje cooperativo en el aula. Barcelona: Paidós.

[3] Turban, E. (1990). Decision support and expert systems: management support systems. Prentice Hall PTR.

[4] Stahl, G. (2011, May). A view of computer-supported collaborative learning research today. In Collaboration Technologies and Systems (CTS), 2011 International Conference on (pp. 325-325). IEEE.

[5] Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. Cambridge handbook of the learning sciences, 2006, 409-426.

[6] Collazos, C., Muñoz, J., & Hernández, Y. (2008). Aprendizaje colaborativo apoyado por computador. In Conferencia Lunes Científico Universidad Militar Nueva Granada,. Bogotá, Colombia.

[7] Stahl, G. (2000). A model of collaborative knowledge-building. In Fourth international conference of the learning sciences (Vol. 10, pp. 70-77). Mahwah, NJ: Erlbaum, 2000a.

[8] Joyanes Aguilar, L. (2003). Fundamentos de programación. Algoritmos, estructuras de datos y objetos. Mac. Graw Hill.

[9] Deitel, H. M., & Deitel, P. J. (2004). Cómo programar en Java. Pearson Educación.

[10]http://roble.pntic.mec.es/jprp0006/tecnologia/bachillerato_tic/unidad01_navegadores/navegadores3.htm (2014)

[11] http://webfoundation.org/about/vision/history-of-the-web/ (2015)

[12] Sommerville, I., & Galipienso, M. I. A. (2005). Ingeniería del software. Pearson Educación.

[13] Li-Li, C., & Zheng-Long, L. (2012, August). Design of Rich Client Web Architecture Based on HTML5. In Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on (pp. 1009-1012). IEEE.

[14] Crowther, R., Lennon, J., Blue, A., & Wanish, G. (2014). HTML5 in Action. Manning.

[15] Leslie, F. (2011). Web standards: mastering HTML5, CSS3, and XML. Apress.

[16] Balasubramanee, V., Wimalasena, C., Singh, R., & Pierce, M. (2013, September). Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development. In Cluster Computing (CLUSTER), 2013 IEEE International Conference on (pp. 1-1). IEEE.

[17] Cochran, D., & Whitley, I. (2014). Bootstrap Site Blueprints. Packt Publishing Ltd.

[18] https://developer.mozilla.org/en-US/docs/Web/JavaScript (2015)

[19] http://www.json.org/ (2015)

[20] Libby, A., & Wellman, D. (2013). JQuery UI 1.10: The User Interface Library for JQuery. Packt Publishing Ltd.

[21] Gackenheimer C. (2013). Node.js Recipes. Apress.

[22] Yaapa, H. (2013). Express web application development. Packt Publishing Ltd.

[23] http://socket.io/f (2015)

[24] Rai, R. (2013). Socket. IO Real-time Web Application Development. Packt Publishing Ltd.

[25] http://www.w3schools.com/svg/default.asp (2015)