UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Departamento de Ingeniería Informática

Proyecto APEP-SDV: Adaptación de plataforma e-learning para personas en situación de discapacidad visual en grado moderado

Rosa Muñoz Valenzuela

Profesor guía: Hector Antillanca

Trabajo de titulación presentado en conformidad con los requisitos para obtener el título de Ingeniero de Ejecución en computacion e Informática

Santiago – Chile
2019

RESUMEN

Palabras Claves:

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1. ANTECEDENTES Y MOTIVACIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

Actualmente la educación tiende hacia el *e-learning* (1), es decir, hacia el aprendizaje en línea. Como apoyo a esta forma de aprendizaje existen diversas plataformas de trabajo siendo una de las más populares en el mundo es Moodle en sus distintas versiones, a nivel mundial la plataforma cuenta con mas de 79 millones de usuarios (2) siendo utilizado en ambientes académicos y empresariales.

La Universidad de Santiago de Chile ofrece herramientas para facilitar la formación de sus alumnos. Una de estas herramientas es la plataforma Moodle en su versión 2.7. Esta permite la creación, administración e intercambio de contenidos virtuales educacionales entre alumnos y docentes. El principal objetivo de la implementación de la plataforma es permitir a los docentes elaborar, compartir, y administrar de forma virtual material académico de relevancia para sus cursos, tales como pruebas, cuestionarios, glosarios, contenidos educativos (3).

La Universidad de Santiago permite el registro e ingreso en sus aulas de cualquier persona que cumpla con los requisitos educacionales exigidos. Incluso ofrece soporte mediante becas y programas de asistencia a quienes presenten capacidades diferentes. En este sentido, la Universidad cuenta con el Programa de acceso inclusivo equidad y permanencia (PAIEP) el cual se encarga de brindar apoyo en el acceso, permanencia y titulación. Los alumnos en situación de discapacidad pueden ingresar en este programa para obtener tutorías y acceso a *hardware* asistivo. Por otro lado, las herramientas y plataformas que la Universidad pone a disposición de su alumnado, también consideran a los alumnos en situación de discapacidad para efectos de usabilidad, siendo Moodle la principal herramienta digital que requiere de accesibilidad.

Moodle es una plataforma del tipo *Learning Content Management System* (LCMS), es decir, permite el manejo, creación y administración de contenidos virtuales . Uno de los grandes desafíos para la plataforma Moodle es implementar la accesibilidad para usuarios en situación de discapacidad. En función de esto, en sus distintas versiones, implementa las normas de accesibilidad web (4) *Web Content Accessibility Guidelines 2.0* (WCAG 2.0) y *Authoring Tool Accessibility Guidelines 2.0* (ATAG 2.0). La accesibilidad web comprende el conjunto de herramientas, sitios y tecnologías web que deben ser adaptadas para su usabilidad por parte de personas en situación de discapacidad (5). En otras palabras, una persona en situación de discapacidad debe ser capaz de percibir, comprender, navegar, interactuar, y finalmente contribuir a la web. Las pautas para la implementación de la accesibilidad web se encuentran formalizadas en la norma *Web Content Accessibility Guidelines 2.0* (WAG 2.0), la que se organiza

en enunciados verificables e independientes de tecnologías, cada uno de los cuales anexa una discusión sobre su implementación en diferentes tecnologías.

Se han detectado áreas de mejora en la accesibilidad para personas en situación de discapacidad visual de la plataforma Moodle 2.7

1.2 PROBLEMA

Los diagnósticos iniciales realizados para determinar el nivel de implementación de medidas de accesibilidad para personas en situación de discapacidad visual muestran que la plataforma Moodle 2.7 implementada por la Universidad de Santiago de Chile no cumple con las medidas establecidas por WCAG 2.0. Por lo tanto, ¿Como brindar acceso a las personas en situación de discapacidad visual en grado moderado a las funcionalidades que implementa Moodle 2.7 ?

1.3 OBJETIVOS

1.3.1 Objetivo general

El principal objetivo es crear una plataforma e-learning LCMS compatible con Moodle 2.7 que implemente medidas de accesibilidad para personas en situación de discapacidad visual en grado moderado.

1.3.2 Objetivos específicos

1. Especificar los requerimientos de la plataforma para el usuario “Alumno USACH en situación de discapacidad visual".
2. Diseñar una plataforma web *e-learning* LCSM accesible para personas en situación de discapacidad visual.
3. Implementar pautas de accesibilidad descritas en la norma WCAG 2.0
4. Implementar compatibilidad con la información de la plataforma moodle 2.7

5. Realizar un diagnostico de la implementación de accesibilidad de la plataforma en desarrollo.

1.4 ALCANCES Y LIMITACIONES

Este trabajo se dedica única y exclusivamente al diseño e implementación de una plataforma web que:

- Implemente las principales operaciones de la plataforma Moodle 2.7, es decir, permita el manejo e intercambio de contenidos con propósitos educacionales.
- Sea compatible con la base de datos relacional implementada en Moodle 2.7
- Soporte navegación por teclado

CAPÍTULO 2. MARCO TEÓRICO

2.1 MARCO TEÓRICO

2.1.1 Accesibilidad web

La primera página web fue publicada el 6 de agosto de 1991 por Tim Berners-Lee en un laboratorio de la Organización Europea para la Investigación Nuclear CERN. La intención de Berners-Lee, inventor de la *World-Wide Web* (WWW) era satisfacer la necesidad de compartir información entre científicos en universidades e institutos alrededor del mundo. Desde entonces, la cantidad de usuarios de Internet alrededor del mundo ha superado los cuatro mil millones de personas (6). Dado el amplio acceso a internet de todo tipo de personas, asegurar que la mayor cantidad de usuarios tengan una buena experiencia de uso de contenidos web se ha convertido en un objetivo deseable en el diseño y construcción de diversas fuentes de información on-line. Esto incluye a personas que posean discapacidades físicas que les impidan interactuar con terminales web tradicionales como computadores o smartphones.

El *World Wide Web Consortium* (W3C), es un consorcio internacional encargado de generar e impulsar recomendaciones y estándares para el crecimiento de la *Word Wide Web*. El W3C propone guías y estándares para todo tipo de actividades relacionadas con Internet, desde manejo de datos y transacciones comerciales on-line, hasta consumo web en vehículos y el Internet de las Cosas.

El 11 de Diciembre de 2008, el W3C puso a disposición del público el documento *Web Content Accesibility Guidelines* (WCAG 2.0) que ofrece lineamiento y recomendaciones en el diseño y elaboración de contenidos web, para su consumo por parte de personas con diversos tipos y grados de discapacidad. La norma WCAG 2.0 establece cuatro principios sobre los cuales se debe erigir toda funcionalidad de accesibilidad en contenidos web:

1. Percibible: el contenido web y sus componentes de interfaz de usuario deben presentarse de forma que el usuario pueda percibirlos. Esto incluye medios alternativos de presentación de información como transcripciones en audio o texto de contenidos.

2. Operable: el contenido web y sus componentes de interfaz de usuario deben ofrecer alternativas de operabilidad para usuarios con capacidades diferentes, de modo que todo el contenido esté disponible para cualquier usuario.

3. Entendible: el contenido web y sus componentes de interfaz de usuario deben ser claros y

entendibles por los usuarios, comportarse de forma predecible, ofrecer ayuda al usuario en caso de errores.

4. Robusto: el contenido web y sus componentes de interfaz de usuarios deben permitir su interpretación mediante tecnologías asistivas de forma confiable y consistente.

2.1.2 Normativa Chilena de Accesibilidad Web

La ley N 20.422, promulgada el 3 de febrero de 2010 por el Congreso Nacional de Chile, establece las normas sobre igualdad de oportunidades e inclusión social para personas en situación de discapacidad en el país. Su principal objetivo es la aseguración del disfrute de derechos y la eliminación de cualquier forma de discriminación fundada en la discapacidad (7).

La ley define la accesibilidad universal como: : “La condición que deben cumplir los entornos, procesos, bienes, productos y servicios, así como los objetos o instrumentos, herramientas y dispositivos, para ser comprensibles, utilizables y practicables por todas las personas, en condiciones de seguridad y comodidad, de la forma más autónoma y natural posible (8)".

Para usos del trabajo de titulo se comprende accesibilidad universal como la condición de ser usable por cualquier individuo sin importar si presenta situación de discapacidad.

El Servicio Nacional de la Discapacidad, desde ahora referenciado como (SENADIS), es una entidad chilena que tiene como principal objetivo promover el derecho a la igualdad de oportunidades de las personas en situación de discapacidad (9). Esta entidad ha definido a la persona en situación de discapacidad como: ’Aquella persona que presenta deficiencias de sus funciones y/o estructuras corporales, limitaciones en sus actividades y restricciones en su participación, como resultado de la interacción negativa de su condición de salud y los factores contextuales (ambientales y personales) en los que se desarrolla’ (10).

El SENADIS establece un índice que cuantifica el máximo nivel posible de funcionamiento que puede alcanzar una persona en un momento dado. Considerando estrictamente su condición de salud, se consideran cuatro grados de discapacidad en una persona:

- Sin discapacidad: La persona no presenta dificultades para llevar a cabo actividades de la vida diaria.
- Discapacidad leve: personas que presentan alguna dificultad para llevar a cabo actividades de la vida diaria, sin embargo la persona es independiente y no requiere apoyo de terceros y puede superar barreras del entorno.

- Discapacidad moderada: personas que presentan una disminución o imposibilidad importante de su capacidad para realizar la mayoría de las actividades de la vida diaria, llegando incluso a requerir apoyo en labores básicas de auto-cuidado y supera con dificultades sólo algunas barreras del entorno.
- Discapacidad severa: personas que ven gravemente dificultada o imposibilitada la realización de sus actividades. cotidiana, requiriendo del apoyo o cuidados de una tercera persona y no logra superar las barreras del entorno.

Para usos estadísticos SENADIS en su Estudio Nacional de la Discapacidad (11) (ENDISC) agrupo las deficiencias funcionales y/o corporales en las categorías presentadas en la figura 4.1.

Categorías de deficiencias en ENDISC Chile 2004	Componentes de la CIF		Categorías en la CIE-10
	Funciones Corporales	Estructuras Corporales	
Físicas	Funciones Neuromusculoesqueléticas y relacionadas con el movimiento	Estructuras relacionadas con el movimiento	Enfermedades del Sistema Osteomuscular y del Tejido Conjuntivo Malformaciones congénitas, deformidades y anomalías cromosómicas Traumatismos, envenenamientos y otras enfermedades del sistema nervioso Afecciones en el período perinatal Causas extremas de morbilidad y de mortalidad (accidentes y otras)
Auditivas	Funciones Sensoriales	El ojo, el oído y estructuras relacionadas	Enfermedades del oído
Visuales			Enfermedades del ojo y sus anexos
Intelectuales[12]	Funciones Mentales	Estructuras del Sistema Nervioso (Estructuras del cerebro)	Trastornos mentales y del comportamiento y Enfermedades del sistema nervioso
Psiquiátricas			
Viscerales [13]	Funciones de los sistemas cardiovascular, hematológico, inmunológico y respiratorio	Estructura de los sistemas cardiovascular, hematológico, inmunológico y respiratorio	Enfermedades del sistema circulatorio Enfermedades de la sangre y de los órganos hematopoyéticos y trastornos de la inmunidad Enfermedades del sistema respiratorio
	Funciones de los sistemas digestivo, metabólico y endocrino	Estructura de los sistemas digestivo, metabólico y endocrino	Enfermedades del sistema digestivo Enfermedades Endocrinas, nutricionales y metabólicas
	Funciones genitourinarias y reproductoras	Estructura genitourinarias y reproductoras	Enfermedades del sistema genitourinario
	Funciones de la piel y estructuras relacionadas	Estructura de la piel y estructuras relacionadas	Enfermedades de la piel y del tejido subcutaneo
Múltiples	Otras Funciones y Estructuras no clasificadas		Otras Enfermedades no clasificadas

Figura 2.1: Clasificación de los tipos de situación de discapacidad ENDISC 2004

El SENADIS ofrece, a través de su página web oficial, un documento denominado Guía de Accesibilidad Web del Servicio Nacional de la Discapacidad. Este documento se basa en la norma WCAG 2.0 del W3C para establecer recomendaciones en la accesibilidad a contenidos web de autoría chilena. Aunque no es obligatorio por ley el seguimiento de este documento, el SENADIS ofrece certificaciones a quienes cumplan con las recomendaciones de este documento, como el Sello Chile Inclusivo que se entrega a : “instituciones públicas y privadas que, sin importar su tamaño, realicen medidas de acción positiva hacia la inclusión social de las personas en

situación de discapacidad".

Tanto la norma WCAG 2.0 como la Guía de Accesibilidad Web del Servicio Nacional de la Discapacidad ofrecen lineamientos para garantizar la accesibilidad a contenidos web de personas con todo tipo de discapacidad físia. Para el desarrollo del proyecto MAVIP, el foco principal se ubica en personas en situación de discapacidad visual.

2.1.3 Heurísticas de Accesibilidad Visual

En su libro *Visual Impairments and Learning*, la doctora Natalie Barraga establece cuatro niveles de discapacidad visual (Barraga, 1992):

1. Discapacidad Visual Moderada: Necesidad de apoyos especiales e iluminación adecuada para realizar tareas visuales, similares a las que podría necesitar una persona con visión normal.

2. Discapacidad Visual Severa: Necesidad de adecuaciones especiales de tiempo y herramientas de apoyo para realizar tareas visuales con errores o inexactitudes.

3. Discapacidad Visual Profunda: Imposibilidad de realizar tareas que requieren de alta agudeza visual, dificultad para realizar tareas que requieren de baja agudeza visual.

4. Ceguera: Imposibilidad de realizar tareas visuales. Percepción de luz sin definición, o carencia de percepción de luz.

Por otro lado, Jakob Nielsen et al. realizaron un estudio de usabilidad web en el año 2001, donde participaron más de 80 usuarios expertos de software, la mayoría de los cuales poseía discapacidades visuales en diferentes niveles. Los resultados de éste y otros estudios similares se encuentran compilados en un documento que ofrece setenta y cinco recomendaciones respecto a la forma en que se despliega la información en una página web. Hoy en día, el Grupo Nielsen-Norman, fundado por Jakob Nielsen, Don Norman y Bruce Tognazzini en 1998, es reconocido por sus estudios en Experiencia de Usuario y Usabilidad en diversos campos tecnológicos, incluyendo y no limitado a la usabilidad web y a la accesibilidad para usuarios con discapacidad.

2.1.4 Moodle

Moodle es un Sistema de Administración de Aprendizaje (*Learning Management System*, LMS) escrito en el lenguaje de programación web PHP y distribuído bajo licencia GNU. Moodle permite la creación de cursos accesibles de forma remota por educadores y alumnos a través de navegadores web, y es utilizado en instituciones educativas alrededor del mundo, incluyendo la Universidad de Santiago de Chile.

Desarrollado originalmente por Martin Dougiamas, la primera versión de Moodle fue lanzada el 20 de agosto de 2002. Hoy en día el Proyecto Moodle es coordinado por la empresa australiana Moodle HQ y apoyado por desarrolladores de código libre alrededor del mundo.

Moodle requiere de un servidor web para ser instalado, así como de una base de datos para almacenar información de cursos y usuarios. Existen versiones autoinstalables de Moodle para sistemas operativos Windows y MacOS con toda la infraestructura de software necesaria preconfigurada. Para distribuciones de Linux existen paquetes precompilados descargables mediante gestores de descarga como APT de Debian.

La última versión estable disponible de Moodle es la 3.5.1, descargable desde la página web del proyecto Moodle. La Universidad de Santiago de Chile, por su parte, ofrece a su alumnado acceso a la plataforma Moodle en su versión 2.7.

2.2 ESTADO DEL ARTE

2.2.1 Diagnostico de accesibilidad Moodle 2.7

Para comprender el estado de la accesibilidad de la plataforma Moodle se la sometió a un test doble de accesibilidad. La primera face consistió en un test automatizado realizado por la herramienta WAVE, la que procesa el código HTML en busca de fallas. La segunda face se realizo en las sesiones presenciales de diseño con el usuario, en estas el usuario mencionaba los elementos que le dificultaban la interacción en las tareas programadas. A continuación se presentan los resultados.

Figura 2.2: Diagnostico Wave parte 1

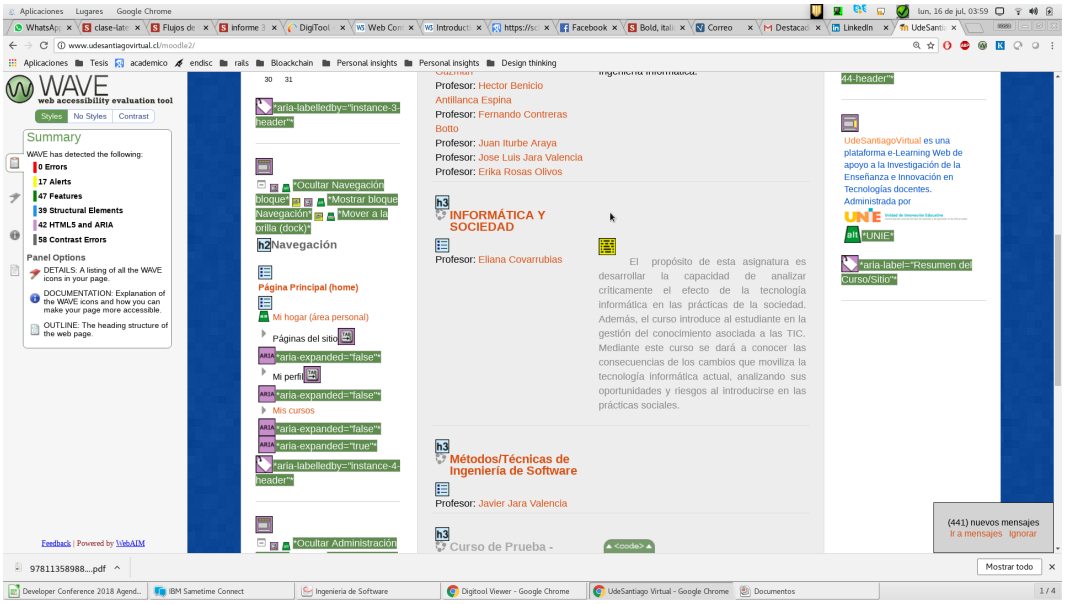

Figura 2.3: Diagnostico Wave parte 2

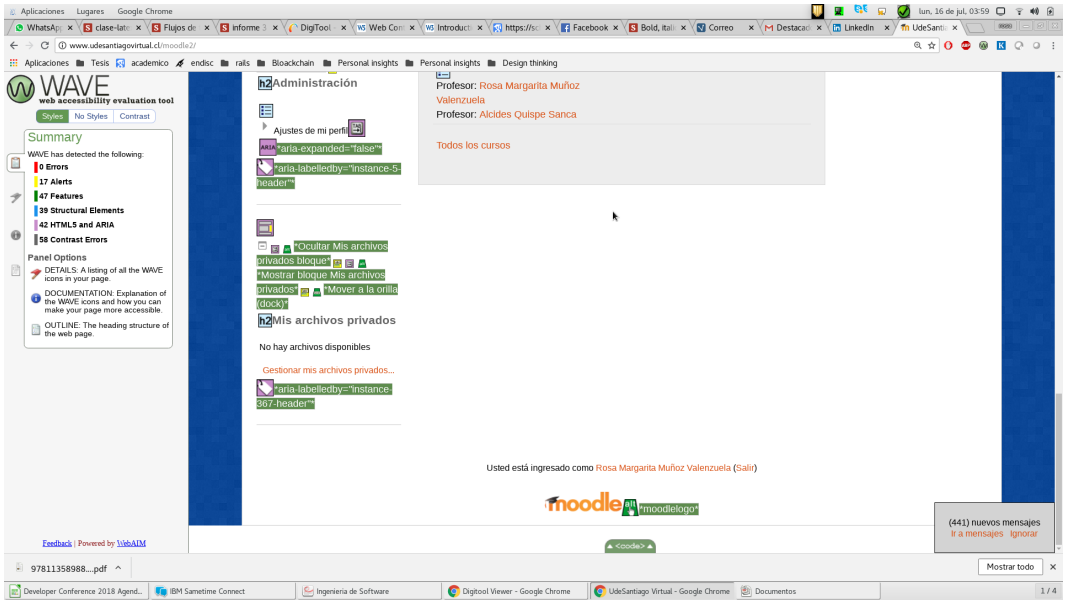

Figura 2.4: Diagnostico Wave parte 3

2.2.2 Diagnostico WAVE

Las figuras 2.2, 2.3 y 2.4 muestran el diagnostico realizado por la herramienta a la plataforma Moodle, a continuación su profundización:

- Errores de contraste: los elementos de la página web no presentan un contraste adecuado, también se debe a que los colores de la página web no son adecuados a los modos de color variable (modo daltónico y de alto contraste)
- Elementos estructurales: se dice de elementos que están ordenados de forma paralela, principalmente elementos que se encuentran listados. Esto dificulta la navegación por teclado.
- HTML 5 y Aria: se presenta el elemento de accesibilidad Aria, el cual consiste en una etiqueta HTML que puede ser leída por lectores de pantalla. La implementación actual es insuficiente según la norma WCAG 2.0.
- Alertas: presenta elementos que poseen leyenda para lectores de pantalla, mas ésta no es el tipo de etiqueta que establece la norma WCAG 2.0

Además se muestran la implementación de treinta y dos características de accesibilidad que corresponden principalmente a la implementación de etiquetas correctas para la descripción de elementos visuales y enlaces.

De lo anterior se desprende que la plataforma Moodle no es accesible para las personas en situación de discapacidad.

2.2.3 Diagnostico con el usuario

Este diagnostico nació de la interacción controlada con la plataforma Moodle 2.7 por parte de un alumno de la Universidad de Santiago de Chile en situación de discapacidad visual, es decir, se le pidió al usuario que hiciera tareas y describiese elementos que le causaran algún tipo de dificultad.

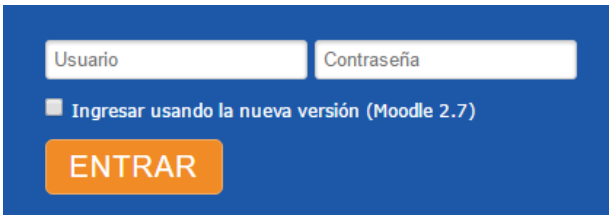

Figura 2.5: Cercanía de elementos de interfaz de usuario.

Incidencia 1 (Figura 2.5): La cercanía de los botones de selección de plataforma y de ingreso a la plataforma hace que sea complejo para el usuario con visión disminuida seleccionar la plataforma sin presionar por error el botón de ingreso.

Figura 2.6: Bajo contraste entre texto y fondo.

Incidencia 2 (Figura 2.6): El contraste del fondo naranjo y las letras blancas hacen que el usuario con visión reducida no pueda distinguir estos elementos. Además, la cercanía de los los elementos hace que sea complejo acceder a estos. Incidencia 3 (Figura 2.7): En la vista

Figura 2.7: Area inaccesible para lectores de pantalla.

de tareas el lector de pantalla no puede acceder a la sección enmarca en rojo, solamente lee los enlaces insertados en esta. Esto provoca que el usuario no comprenda el contenido de la vista.

Figura 2.8: Enlaces inaccesibles para lector de pantalla.

Incidencia 4 (Figura 2.8): El lector de pantalla no puede acceder a la región del mensaje escrito, mas puede acceder a los vínculos insertos en éste.

Incidencia 5 (Figura 2.9): Los iconos Nuevo Mensaje, “Bloquear Contacto” e “Historial de Mensajes”, se encuentran muy cercanos entre sí, lo que causa la selección errónea por confusión visual en los usuarios con visión disminuida.

Incidencia 6 (Figura 2.10): En la vista de novedades, si no existen elementos el lector de pantalla no ingresa en el recuadro gris que indica la inexistencia de novedades.

Incidencia 7 (Figura 2.11): El uso de menús desplegables dificulta la navegación para usuarios con visibilidad reducida dado que al hacer incremento de la pantalla (zoom) se excede lo que se puede mostrar en la pantalla, y al bajar para ver las opciones desplegadas estas se cierran

Incidencia 8 (Figura 2.12): Bajo contraste entre ícono y fondo hace que el elemento marcado sea difícil de ver. Además, al hacer zoom en la pantalla el elemento queda demasiado cerca de los enlaces en naranjo y es difícil seleccionar el elemento correcto.

2.3 ESTUDIOS SOBRE LA ACCESIBILIDAD DE LA PLATAFORMA MOODLE

La plataforma Moodle esta en continuo desarrollo ((12)). Este se basa en en la retroalimentación de usuarios y el desarrollo constante. Los usuarios notifican errores, los que

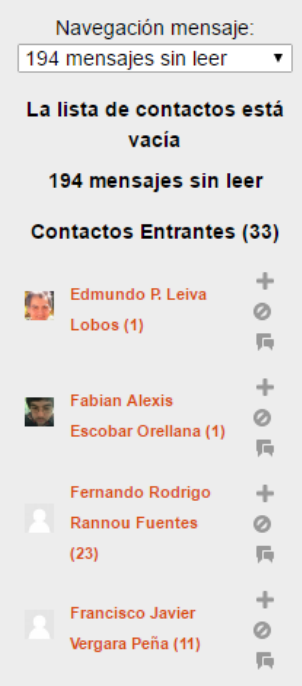

Figura 2.9: Iconos muy cercanos entre sí.

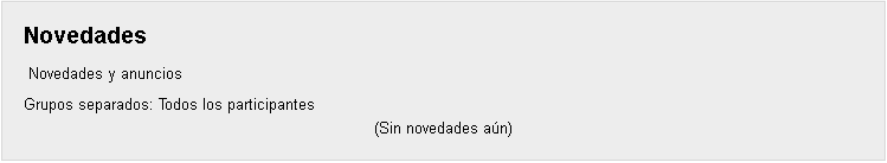

Figura 2.10: Inaccesibilidad por carencia de elementos.

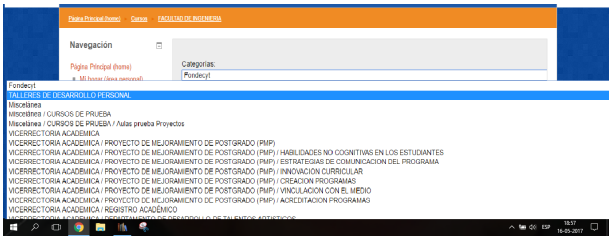

Figura 2.11: Inaccesibilidad de menú desplegable con pantalla aumentada.

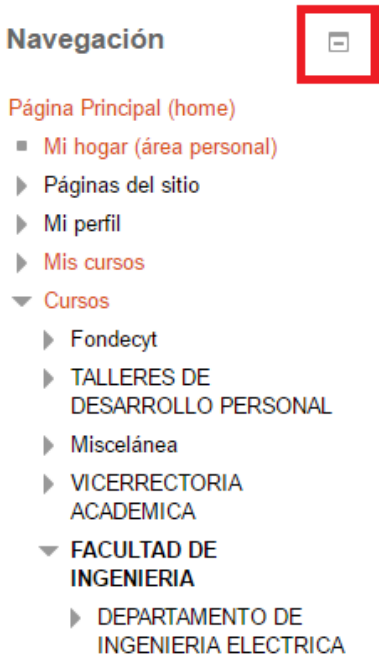

Figura 2.12: Bajo contraste y cercanía con otros elementos en pantalla.

son corregidos por los desarrolladores e integrados en las nuevas versiones. Por parte de los desarrolladores estos realizan nuevos módulos para la plataforma los que incorporan nuevas funcionalidades.

La plataforma posee funciones de accesibilidad, más no existe documentación en la plataforma que indique el grado de accesibilidad de esta (4). Diversos autores se han dedicado al diagnóstico de la accesibilidad de la plataforma Moodle, a continuación se muestran algunos.

En el 2011 Calvo, Iglesias y Moreno (13) realizan un diagnóstico a la usabilidad de la plataforma Moodle para personas en situación de discapacidad visual. Los resultados de dicho diagnóstico fueron que la plataforma Moodle en su versión 1.6, no cumple en su totalidad con las normas *Authoring Tool Accessibility Guidelines 1.0* (14) y *Web Content Accessibility Guidelines 2.0* (15). Las principales falencias encontradas problemas con la representación con lectores de pantalla y el acceso a contenidos. El enfoque principal de este trabajo es el diagnóstico de la plataforma Moodle, este enfoque no es compatible con la solución planteada dado que esta considera la modificación de la plataforma para cumplir con la normativa vigente (WCAG 2.0).

Los autores Calvo, Iglesias y Moreno (16) realizan un segundo diagnóstico a la usabilidad de la plataforma Moodle para personas en situación de discapacidad. En este segundo diagnóstico se centrarón en la versión 2.0 de la plataforma, abordan la creación y subida de material a la plataforma, lo que verifica el cumplimiento de la norma ATAG 2.0 (17). El diagnóstico tiene como resultado que si bien la plataforma cumple con la norma en su grado mínimo, la persona en situación de discapacidad visual aun no puede subir contenido a la plataforma sin

asistencia de terceros. Como en su anterior trabajo el enfoque central es el diagnóstico, lo que no es compatible con la solución planteada.

Los autores Castaño, Iglesias y Calvo (18) postulan que las aplicaciones de chat pertenecientes a las plataformas *e-learning* tienen un alto impacto en el aprendizaje de los alumnos. Dado esto evalúan la accesibilidad de personas en situación de discapacidad en las plataformas Moodle 2.7, Edmodo e Instructur. El resultado de esta evaluación es que todas las plataformas presentan falencias según la norma WCAG 2.0, es la plataforma Moodle 2.7 la que implementa mas medidas de accesibilidad entre las plataformas evaluadas. El enfoque presentado en este trabajo se diferencia del enfoque de la solución propuesta en dos puntos, el primero, las situaciones de discapacidad que abarca, para la solución solo se aborda la situación de discapacidad visual en sus grados severo y moderado, el segundo punto es el aspecto y plataformas a evaluar, lo que distan radicalmente del enfoque de la solución.

El autor Kumar realiza un diagnóstico de accesibilidad de distintas plataformas empleadas para el estudio por estudiantes que posean o no situación de discapacidad (19). Las plataformas evaluadas fueron Youtube, Moodle 2.0, Presentaciones Power Point y otras paginas web, dichas plataformas se examinaron a la luz de la norma WCAG 2.0. El estudio demostró que las plataformas no cumplían con la norma, además, su diseño no estaba centrado en el estudiante, y que esto era vital para el mejoramiento de la accesibilidad. El enfoque de este estudio dista del enfoque de la solución propuesta dado que este no acota la situación de discapacidad, además no solo examina la plataforma Moodle 2.0 sino otras plataformas que no están especializadas en *e-learning*.

Dado los antecedentes presentados y el diagnóstico realizado se estable que el enfoque de la solución propuesta, es decir, la creación de un sistema alternativo a la plataforma Moodle en su versión 2.7 para proporcionar accesibilidad a personas en situación de discapacidad en sus grados severos y moderado cuenta con precedentes presentados en la literatura y que aun no han sido solucionados.

2.4 ENFOQUES DE LA SOLUCIÓN

El SENADIS define a las tecnologías para la inclusión de personas en situación de discapacidad como: : “Cualquier dispositivo, software, equipo, sistema o instrumento fabricado, desarrollado o adaptado que permitan superar y/o eliminar las barreras arquitectónicas, actitudinales y procedimentales que enfrentan las personas en situación de discapacidad durante su desempeño cotidiano"(20).

Dicha definición abarca los dispositivos que permitan sortear barreras del entorno a

las personas en situación de discapacidad. Esto incluye al hardware y software que asistan a las personas en situación de discapacidad, los que se definen como (21):

- **Hardware asistivo:** Cualquier producto, instrumento, equipo o sistema técnico utilizado por una persona en situación de discapacidad, fabricado especialmente o disponible en el mercado para prevenir, compensar, mitigar o neutralizar la deficiencia, incapacidad o discapacidad en esta gama se presentan dispositivos varios tales como teclados braille, apuntadores, emuladores de teclado o mouse.
- **Software asistivo:** se define como la disciplina de la ingeniería que se ocupa de todos los aspectos de la producción de software, esta posee actividades fundamentales para la producción, estas son: especificación, desarrollo, validación y evolución, bajo este ciclo genérico se abarca todo el proceso de software siendo amplia y general.

2.5 JUSTIFICACIÓN DEL ENFOQUE SELECCIONADO

La solución propuesta se enfocará en el software asistido. Esto se debe a que la plataforma Moodle es una plataforma de software y no presenta hardware asistivo especializado asociado, es decir, si bien la plataforma Moodle soporta el hardware asistivo, este no requiere un hardware asistivo especifico.

CAPÍTULO 3. METODOLOGÍA, HERRAMIENTAS Y AMBIENTE DE DESARROLLO

3.1 METODOLOGÍA DE DESARROLLO

El proyecto aborda un proceso de negocios conocido como es la plataforma Moodle de apoyo al estudiante, y la cohesiona con el enfoque hacia la usabilidad para usuarios con discapacidad visual. Este nuevo enfoque es desconocido para la desarrolladora del proyecto, y añade un componente de incertidumbre a la definición de los requerimientos del sistema. Afortunadamente se cuenta con una alta disponibilidad de un grupo de usuarios con tales discapacidades, quienes pueden ser considerados *stakeholders* del proyecto y proponen modificaciones y nuevas funcionalidades en base a sus requerimientos.

Por otro lado, existe una alta incertidumbre respecto del grado de integración del proyecto con la base de datos de la plataforma Moodle 2.7, dado que la plataforma debe coexistir con Moodle 2.7 y la estructura de la base de datos compartida no puede ser alterada.

Dadas las fuentes de incertidumbre definidas anteriormente y la disponibilidad de usuarios *stakeholders*, se decide utilizar las metodologías *Rapid Application Development* (RAD) y *Extreme Programming* (XP).

La metodología RAD se basa en el concepto de prototipo descartable (22), artefacto compuesto por una aplicación prototipo funcional descartable que es validada por los *stakeholders* y mejorada en iteraciones de desarrollo, con lo que es posible evaluar múltiples opciones de satisfacción de requerimientos de forma práctica.

En pos de tomar ventaja del alto *feedback* brindado por el usuario final para mitigar la incertidumbre durante el proceso de desarrollo, se realiza una primera etapa de especificación de diseño y maquetas de interfaz gráfica funcionales en compañía del usuario, utilizando para ello la metodología RAD. Esto incluyo el estudio de las normas de accesibilidad para definir un marco de trabajo común.

El producto de la primera etapa de desarrollo guía el desarrollo de la aplicación final utilizando metodología XP, metodología que permite el desarrollo incremental de las funcionalidades definidas por los *stakeholders*, quienes continúan entregando *feedback* sobre el proyecto en desarrollo.

Cada reunión con los *stakeholders* constituye la entrega de una nueva funcionalidad, la cual puede ser aceptada o modificada en sus características. Durante cada reunión con los clientes se realizan exploraciones guiadas de las características desarrolladas, se recopila una lista de modificaciones y nuevas características deseadas, y se comprometen las metas a cumplir

hasta la siguiente reunión.

Durante la fase desarrollo se emplean historias de usuario para determinar los requisitos funcionales y no funcionales que debe cumplir el producto final. El test de validación de cada historia consiste en la exploración de la funcionalidad con el usuario esperando que éste se encuentre satisfecho en cuanto a su experiencia de uso, y recolectando una lista de incidencias en caso de que la prueba no sea exitosa.

Tanto en la fase de diseño inicial como en la de desarrollo, se contó con la ayuda de un segundo programador que apoyó la evolución del proyecto según los requerimientos de cada metodología de trabajo: realizando pruebas de calidad durante ambas fases y sugiriendo funcionalidades y pruebas de aceptación para cada reunión con los clientes.

Para controlar las tareas que se realizan durante el desarrollo del proyecto se utiliza un tablero de tareas inspirado en la metodología Kanban. Este registro permite controlar el nivel de avance del proyecto, así como el nivel de completitud de las tareas que constituyen la satisfacción de los requerimientos manifestados por los usuarios *stakeholders* tanto al inicio del proyecto como tras cada reunión de análisis y *feedback* de la herramienta de *software*.

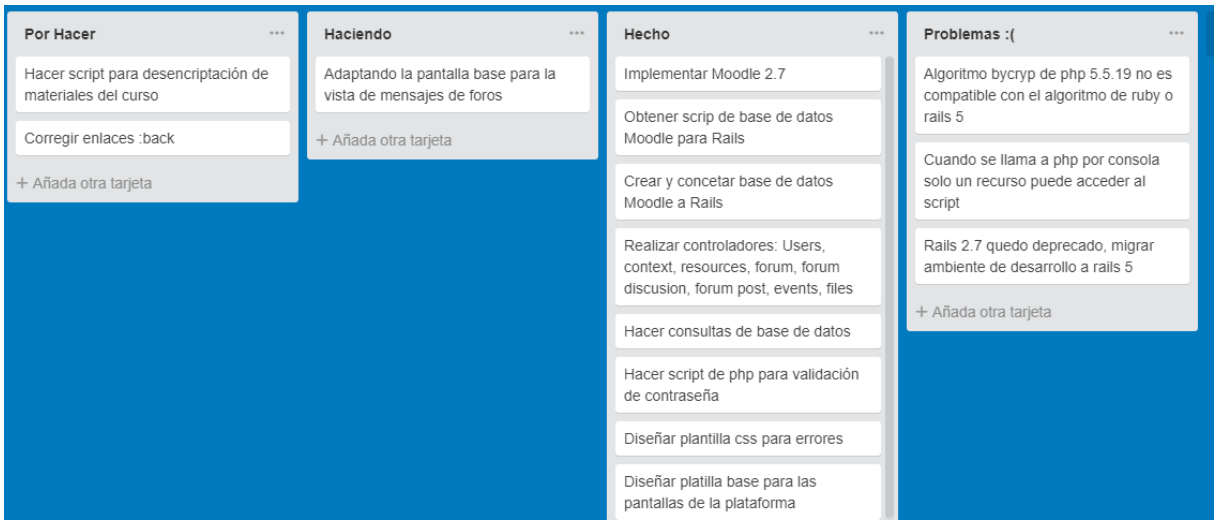

Figura 3.1: Tablero Kanban del proyecto

3.2 HERRAMIENTAS DE DESARROLLO

Para el desarrollo se utilizarán las siguientes herramientas de software:

- Sistema Operativo Ubuntu 16.04 LTS.
- Editor de texto Atom, para desarrollo de Ruby on Rails.

- Plataforma Moodle 2.7.
- Pata levantar la plataforma moodle 2.7 se uso el stack xampp 5.5.19 que tiene los siguientes componentes:
  - PHP 5.5.19
  - Mysql 5.0.11
  - Apache 2.4.10
- Para la construcción dela plataforma se empleo Ruby 2.52 y Rails 5.2
- Navegador Google Chrome Versión 54.0 para pruebas.
- Sharelatex, para documentos y memoria.
- Trello, para tablero Kanban.
- Wave para pruebas automatizadas de accesibilidad.
- MySql Workbench para modificación de base de datos y realización de consultas.
- Plataforma Heroku, para alojamiento de servidor de pruebas y producción.

Respecto al hardware, se utilizará el siguiente dispositivo:

- Notebook HP Probook 655, el cual cuenta con 8 Gigabytes de memoria ram, procesador Amd A8-5550AM. Este equipo será empleado para desarrollo, pruebas, tanto unitarias como de módulo y aceptación, levantamiento de servidor y despliegue.

3.3 AMBIENTE DE DESARROLLO

El ambiente de desarrollo corresponderá a las dependencias del Departamento de Ingeniería Informática de la Universidad de Santiago de Chile, lugar donde se hará uso de los recursos de impresión, biblioteca y la red de Internet.

CAPÍTULO 4. REQUERIMIENTOS

La definición de los usuarios y de sus necesidades constituyen las bases fundamentales del proceso de diseño del software. El proceso de levantamiento de requerimientos ayuda a esclarecer esta definición a través de la definición del negocio y de sus usuarios involucrados.

La plataforma se enfoca en persona en situación de discapacidad visual en sus grados severo y medio. Esto comprende específicamente personas con perdida total o parcial del sentido de la vista y personas con defectos en la visión de color, en el anexo A se incluye información específica de los defectos de visión de color que se incluyen.

4.1 PLATAFORMA MOODLE

El proyecto MAVIP en desarrollo busca apoyar el funcionamiento de la plataforma existente actualmente, la cual se basa en el software Moodle 2.7. Esta herramienta se encuentra en etapa de producción, es decir, se encuentra disponible para su uso regular por parte del alumnado de la Universidad de Santiago de Chile, y el proyecto MAVIP debe coexistir con ella para entregar un canal accesible de uso para alumnos con discapacidad visual.

La coexistencia con la plataforma Moodle 2.7 impone restricciones en el diseño y construcción del proyecto MAVIP debido princpalmente a que ambas herramientas se alimentan de la misma base de datos, la cual no puede ser modificada. MAVIP debe adaptarse a la estructura de base de datos utilizada por Moodle, utilizar las mismas estructuras de datos, y adaptar sus funciones según los flujos de información de Moodle 2.7. Desde el punto de vista de la información almacenada en la base de datos, desde credenciales de usuario hasta material académico y mensajería entre usuarios, un usuario Moodle debe ser indistinguible de uno MAVIP.

4.2 NORMAS DE ACCESIBILIDAD

4.2.1 Norma WAG 2.0

A continuación se presenta un extracto de la norma WCAG 2.0 ((15)), las pautas presentadas a continuación son la principal guía para el diseño de la plataforma.

- Contenido no textual: Todo contenido no textual que se presenta al usuario cuenta con una alternativa textual que sirve para un propósito equivalente.
- Adaptabilidad: Cree contenidos que puedan presentarse de diversas maneras (como por ejemplo una composición más simple) sin perder la información ni su estructura
- Distinguible: Haga más fácil para los usuarios ver y oír el contenido, incluyendo la separación entre primer plano y fondo.
- Accesible a través del teclado: Haga que toda funcionalidad esté disponible a través del teclado.
- Ataques: No diseñe un contenido de manera que se sepa que puede causar ataques.
- Ayuda a la entrada de datos: Ayude a los usuarios a evitar y corregir errores.

4.3 DIRECTRICES DE ACCESIBILIDAD

A continuación se presenta una traducción literal de algunas de las directrices de accesibilidad desarrolladas por *The Nielsen group* (23). Las directrices que se presentan poseen mayor afinidad con el propósito de la plataforma.

- Minimiza el uso de elementos gráficos
- Evite usar ventanas emergentes
- Evite abrir ventanas emergentes en el navegador
- No confíe en el texto de transferencia (textos emergentes) para transmitir cualquier información
- Evite el uso de menús cascada
- Limite el numero de enlaces por pagina
- Evite botones muy pequeños y texto pequeño para enlaces
- Deje espacio entre enlaces y botones.
- Evite usar imágenes como el único método para vincular a algo.
- Subraya todos los enlaces.

- Crear enlaces dentro del texto cuando tenga sentido. Use solo botones adicionales cuando es necesario.
- Confirme de inmediato el nombre de la página (empresa) una vez que la página principal se haya cargado.
- Minimice la necesidad de desplazarse.
- Cuando los usuarios deben hacer una elección, mantenga todas las posibilidades en la misma vecindad.
- Cuando los usuarios deben hacer una elección, advertirles que la elección está por venir, y dígales cuántas opciones tienen.
- Diseña páginas consistentemente.
- Limite la cantidad de información que requieren los formularios; recoger solo el mínimo necesario.
- No use solo texto rojo o resaltado amarillo para indicar errores.
- Asegúrese de que el orden de las pestañas sea lógico.
- En formularios, coloque el botón Enviar lo más cerca posible de la última entrada de campo cuadro o herramienta de selección en el formulario.
- Elija colores de texto para un buen contraste.
- No use texto muy pequeño para el texto del cuerpo.
- No use encabezados y categorías de texto pequeños o sutiles.
- Siempre cree un buen contraste entre el texto y el fondo de la página.
- No confíes en una imagen de fondo como fondo de página para crear contraste con el texto.
- Pruebe las fuentes y colores de texto de su sitio con ampliadores de pantalla.
- Asegúrese de que es posible ampliar su sitio.
- Escriba concisamente y elimine el texto superfluo.

4.4 ESPECIFICACIÓN DE REQUERIMIENTOS

Para el desarrollo de la plataforma en base a las normas vigentes y las necesidades del alumno se realiza la especificación de requerimientos. Los requerimientos puedes ser de dos tipos; funcionales o no funcionales (24).

Los requisitos funcionales definen los componentes y funciones del sistema de software. La propuesta MAVIP apunta a conservar los requisitos funcionales más importantes de la plataforma Moodle según sus usuarios, a la vez que añadiendo requisitos funcionales dirigidos a la usabilidad de la plataforma para usuarios con discapacidad visual.

Los requisitos no funcionales describen propiedades del sistema que no corresponden a componentes ni funciones, sino a parámetros medibles de características y calidad tales como rendimiento, seguridad, disponibilidad, etc. Estos requisitos definen los alcances y limitaciones del sistema MAVIP.

4.4.1 Requerimientos funcionales

- RF 01: El sistema debe informar al usuario en una vista diferente el ingreso de un valor erróneo.
- RF 02: El sistema debe permitir al usuario seleccionar a lo menos una opción de redirección cuando la sección este vacía.
- RF 03: El sistema debe desplegar una lista de los cursos inscritos por el usuario.
- RF 04: El sistema brindara la opción de buscar cursos.
- RF 05: El sistema debe permitir la búsqueda de cursos dado su nombre o parte de este.
- RF 06: El sistema debe permitir la inscripción de cursos mediante el ingreso de contraseña.
- RF 07: El sistema debe desplegar una lista de los materiales disponibles por cada curso.
- RF 08: El sistema debe trasferir archivos dada la solicitud del usuario.
- RF 09: El sistema debe informar el inicio de la transferencia del archivo
- RF 10: El sistema desplegará una lista de todos los eventos disponibles en el curso.
- RF 11: El sistema debe soportar la carga de archivos para un evento.
- RF 12: El sistema debe desplegar una lista de foros disponible en el curso

- RF 13: El sistema debe desplegar una lista de las discusiones disponibles en cada foro.
- RF 14: El sistema debe mostrar el mensaje principal de una discusión.
- RF 15: El sistema debe desplegar una lista con todos los mensajes secundarios de una discusión.
- RF 16: El sistema debe permitir la opción de crear un nuevo mensaje secundario dentro de una discusión.

4.4.2 Requerimientos no funcionales

- RNF 01: La interfaz gráfica del sistema debe soportar un zoom de hasta un 200 % sin perder su diseño.
- RNF 02: La interfaz gráfica del sistema debe tener colores distinguibles por los usuario con deficiencias en la visión y con baja visión
- RNF 03: La interfaz gráfica del sistema debe soportar navegación por teclado
- RNF 04: La interfaz gráfica del sistema debe usar elementos que interactúen con la navegación por teclado
- RNF 05: La interfaz gráfica debe poseer colores contrastantes
- RNF 06: La interfaz gráfica debe poseer una tipografía distinguible y ampliable.

4.5 PROCESO DE DISEÑO

El proceso de diseño de la plataforma puede ser dividido en dos grandes aspectos, estos son diseño de operaciones, en el que se definieron las interacciones y flujo de tareas de la plataforma, y diseño de interfaz se definieron los elementos gráficos a usar y se adecuaron al usuario.

4.5.1 Diseño de operaciones

4.5.2 Diseño de interfaz

En esta etapa se obtuvo la ayuda de una alumno de la Universidad de Santiago de Chile el cual se encuentra en situación de discapacidad visual en su grado moderado. El usuario presenta una enfermedad degenerativa que deriva en la perdida de visión progresiva.

El proceso de diseño con el usuario fue iterativo mediante sesiones. Se realizaron 3 sesiones de diseño presenciales y 5 sesiones virtuales. Las sesiones de diseño presenciales consistieron en interacciones controladas con el usuario y la plataforma Moodle 2.7 implementada por la universidad. El objetivo principal de estas sesiones era observar las tareas y elementos dificultaban la interacción del usuario y la plataforma. Las sesiones de diseño virtual consistían en propuestas de interfaz para la corrección del usuario. A continuación se presentan las maquetas de la interfaz gráfica.

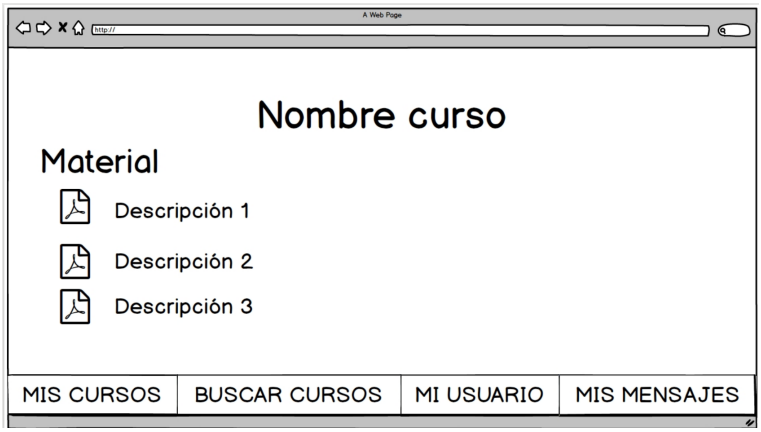

Figura 4.1: Primera sesión de diseño virtual de la interfaz gráfica

Dado el diagnostico previo de la interfaz de usuario y el *feedback* del alumno, se detecto que los múltiples elementos de presentados en una sola pantalla de la plataforma entorpecen tanto el foco del usuario un una tarea como la selección de elementos dado la cercanía de estos. En consecuencia se decidió separar las operaciones de la plataforma en secciones especificas. Cada sección solo admite solo una tarea específica para un elemento. En la figura 4.2 muestra la sección de materiales de un curso en la cual seleccionando un elemento puedes descargarlo, se puede apreciar que solo se incluyen las enlaces e iconos a los materiales del curso más información contextual (nombre del curso e identificador de la sección) más una barra de navegación al fondo para dirigirse a operaciones rápidas dentro de la plataforma.

Al usuario le agrado la separación de las secciones de la plataforma, esto pe permitió

enfocarse en una tarea específica por vez además de facilitar la selección de elementos. Más los iconos que acompañaban a los enlaces se le dificultaban la interacción dado que el usuario los percibía borrosos o como una mancha negra dentro de la pantalla. Dado esto se paso a la segunda iteración.

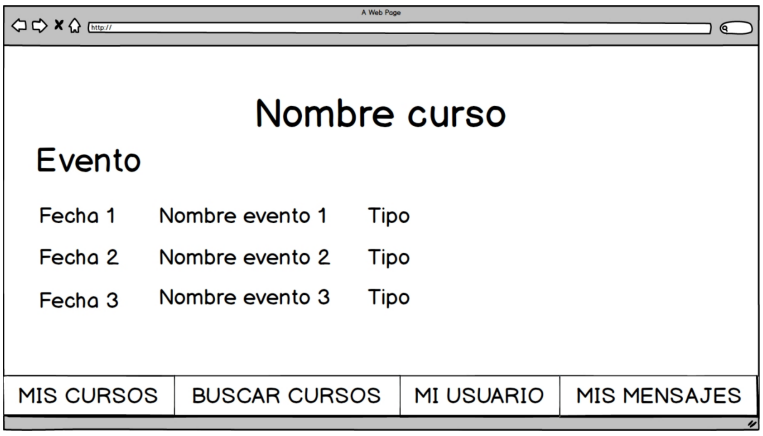

Figura 4.2: Segunda sesión de diseño virtual de la interfaz gráfica

En la figura 4.3 se puede apreciar la sección de eventos de la segunda etapa de diseño, en esta se eliminaron los iconos que acompañaban a los enlaces dejando solamente enlaces descriptivos del elemento. Se conservo la separación en secciones. Con respecto a esto el usuario cree que la interfaz se veía mas clara que que en el diseño previo pero es compleja la selección de un enlace dado que no estaban delimitados. Dado esto se paso a la tercera iteración.

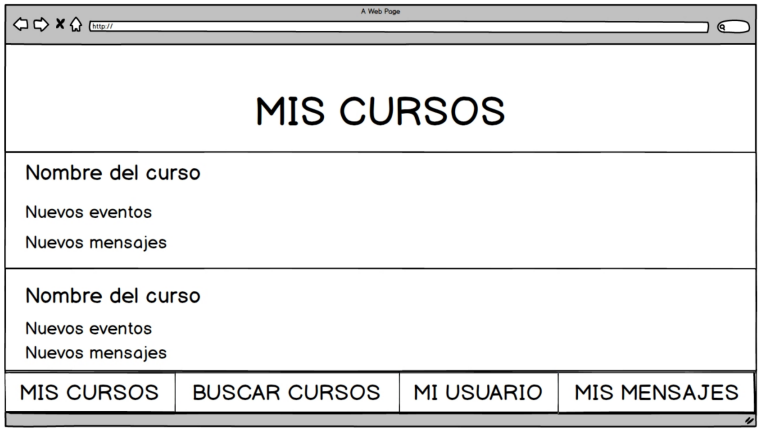

Figura 4.3: Tercera sesión de diseño virtual de la interfaz gráfica

En la figura 4.4 se puede apreciar la sección de cursos de la tercera etapa de diseño, en esta se decidió reemplazar los enlaces por botones largos, además se modifico la sección de cursos para mostrar un acceso directo a las secciones principales del curso, se mantuvo el diseño por secciones. Con respecto a esto el usuario creo que el diseño con botones le ayuda a

seleccionar los elementos de la pantalla, En específico con la modificación de cursos con accesos directos se descarta de inmediato, dado que al tener mas de un enlace por botón se vuelve complejo tanto de leer como seleccionar. En función de esto se pasa a la cuarta iteración.

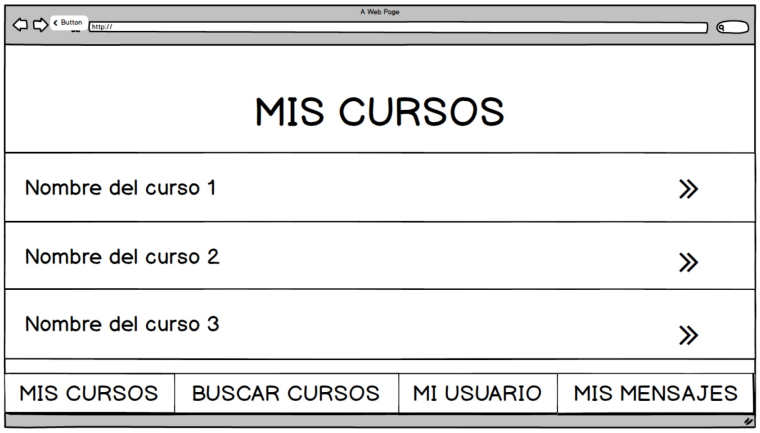

Figura 4.4: Cuarta sesión de diseño virtual de la interfaz gráfica

En la figura 4.5 se puede aprecia la sección de cursos de la cuarta etapa de diseño, en esta se decidió eliminar los enlaces mutiles dentro del botón de cursos pasando a un botón largo. El usuario está satisfecho con la plataforma, más dentro de la misma sesión de diseño se denoto que no existía forma de pasar de una sección a otra, en función de esto se pasa a la quinta iteración de diseño.

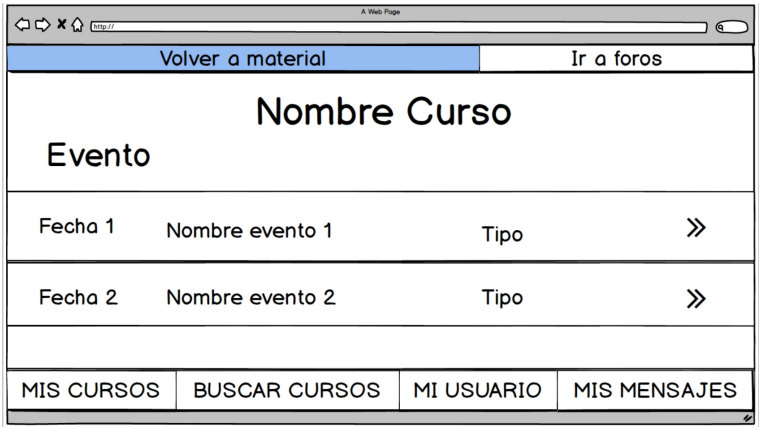

Figura 4.5: Quinta sesión de diseño virtual de la interfaz gráfica

En la figura 4.6 se puede apreciar la sección de eventos de la plataforma, se agrego una barra de navegación que permite avanzar entre las secciones. Sin más comentarios por parte del usuario se valida este diseño de interfaz como base para la implementación de plataforma. En el anexo B se puede encontrar el diseño completo de la interfaz.

CAPÍTULO 5. CONCLUSIONES

GLOSARIO

Aprendizaje Profundo (*Deep Learning*): Término utilizado en el contexto de la investigación sobre Redes Neuronales para referirse a modelos complejos con múltiples capas neuronales como redes Convolucionales, LSTM, entre otros.

nVidia CUDA: Plataforma de computación paralela y modelo de programación desarrollado por nVidia. Permite el uso de GPUs compatibles para realizar cálculos matemáticos a mayor velocidad que con CPUs convencionales. CUDA posee múltiples aplicaciones en áreas como modelamiento de procesos mecánicos y físicos, procesamiento de *Big Data*, y entrenamiento de Redes Neuronales para *Deep Learning*, entre otros.

Dispositivo de Bajo Poder de Cómputo: Dispositivo computacional que, por sus características de capacidad de procesamiento, memoria interna y/o espacio de almacenamiento, no es capaz de hospedar ni ejecutar una Red Neuronal tradicional. Algunos dispositivos que coinciden con esta descripción corresponden a *smartphones* de bajo costo, dispositivos del Internet de las Cosas, Computadores de Una Placa (*Single-Board Computers*) como Raspberry Pi, etc.

Dispositivo de Alto Poder de Cómputo: Dispositivo computacional que, por sus características de capacidad de procesamiento, memoria interna y/o espacio de almacenamiento, tiene la capacidad de hospedar y ejecutar una Red Neuronal tradicional. Algunos dispositivos que coinciden con esta descripción corresponden a smartphones de gama alta, computadores de escritorio con CPUs multinúcleo o tarjetas de video compatibles con CUDA, servidores, etc.

Red Neuronal Pequeña (RNP): Red Neuronal alojada en un dispositivo de bajo poder de cómputo, conectada de forma remota con una Red Neuronal Grande (RNG) como parte de la arquitectura *Big-Little*. La Red Neuronal Pequeña se modela en base a la Red Neuronal Grande, y clasifica un subconjunto de clases consideradas “críticas” para la tarea de clasificación. Si la Red Neuronal Grande no puede clasificar exitosamente una muestra de datos, la transmite a la Red Neuronal Grande para que realice la clasificación. Dado que la Red Neuronal Pequeña está diseñada para estar físicamente cerca de la fuente de datos a clasificar, la arquitectura puede encontrar una clasificación en menor tiempo su la Red Neuronal Pequeña es la que realiza la clasificación.

Red Neuronal Grande (RNG): Red Neuronal alojada en un dispositivo de alto poder de cómputo, conectada de forma remota con una Red Neuronal Pequeña (RNP) como parte de la arquitectura *Big-Little*. La Red Neuronal Grande es diseñada para clasificar todas las clases de datos posibles para un problema de clasificación dado, pero sólo realiza clasificaciones si su Red Neuronal Pequeña asociada no puede hacerlo. Dado que la Red Neuronal Grande está diseñada para encontrarse físicamente lejos de la fuente de datos a clasificar, la arquitectura sólo la utiliza para datos no “críticos” que, para el problema de clasificación dado, pueden tomar más tiempo en ser clasificados.

Nodo: Dado que las arquitecturas *Big-Little* y Red de Redes Neuronales están compuestos por múltiples Redes interconectadas, un Nodo corresponde a una de estas Redes Neuronales. En el caso de *Big-Little*, sus nodos son Red Neuronal Pequeña (RNP) y Red Neuronal Grande (RNG), mientras que en el caso de Red de Redes Neuronales, cada Red Neuronal recibe como apellido su posición en el proceso lineal de clasificación, siendo el Nodo 1 la red neuronal que recibe la muestra de datos a clasificar, y el Nodo N la red que, sin importar su resultado de clasificación, notifica al Nodo N que el proceso ha alcanzado su máxima posible extensión.

BIBLIOGRAFÍA

[1] M. J. Rosenberg, *E-learning: Strategies for delivering knowledge in the digital age*. McGraw-Hill New York, 2001, vol. 3.

[2] Moodle, “Acerca de moodle.” [Online]. Available: https://docs.moodle.org/all/es/Acerca_de_Moodle

[3] U. de innovación educativa (UNIE USACH), “Importante: Desde ahora los docentes pueden preparar sus asignaturas 2014 en el sistema moodle,” nov 2103.

[4] Moodle, “Accesibilidad - moodledocs.” [Online]. Available: https://docs.moodle.org/all/es/Accesibilidad#ARIA_1.0

[5] W. W. W. Consortium *et al.*, “Web content accessibility guidelines (wcag) 2.0,” 2008.

[6] I. W. Stats, “Internet usage statistics - the internet big picture,” 2018.

[7] C. Nacional, “Ley 20.422.” [Online]. Available: Navegar

[8] ——, “Ley 20.422.” [Online]. Available: Navegar

[9] S. nacional para la discapacidad, “¿quienes somos?”

[10] S. nacional para la discapacidad SENADIS e Intituto Nacional de estadística INE, “Estudio nacional de la discacidad (endisc),” pp. 37–40, 2014.

[11] S. nacional para la discapacidad SENADIS e Instituto Nacional de estadística INE, “Estudio nacional de la discapacidad (endisc),” pp. 59–60, 2004.

[12] Moodle, “Desarrollo - moodledocs.” [Online]. Available: https://docs.moodle.org/dev/Tracker_guide#Tracker_groups_and_permissions

[13] Calvo, Iglesias, and Moreno, “Is moodle accessible for visually impaired people?” in *International Conference on Web Information Systems and Technologies*. Springer, 2011, pp. 207–220.

[14] W3C, “Authoring tool accessibility guidelines 2.0,” "World Wide Web Consortium". [Online]. Available: https://www.w3.org/TR/2000/REC-ATAG10-20000203//

[15] ——, “Web content accessibility guidelines 2.0,” "World Wide Web Consortium". [Online]. Available: https://www.w3.org/TR/WCAG20/

[16] Calvo, Iglesias, and Moreno, “Accessibility barriers for users of screen readers in the moodle learning content management system,” *Universal Access in the Information Society*, vol. 13, no. 3, pp. 315–327, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10209-013-0314-3

[17] W3C, “Authoring tool accessibility guidelines 2.0,” "World Wide Web Consortium". [Online]. Available: https://www.w3.org/TR/ATAG20/

[18] Calvo, Iglesias, and Castaño, “Evaluation of accessibility barriers and learning features in m-learning chat applications for users with disabilities,” *Universal Access in the Information Society*, pp. 1–15, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10209-016-0484-x

[19] R. Kumar, Kari L.and Owston, “Evaluating e-learning accessibility by automated and student-centered methods,” *Educational Technology Research and Development*, vol. 64, no. 2, pp. 263–283, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11423-015-9413-6

[20] SENADIS, “Tecnologías para la inclusión educativa de personas en situación de discapacidad,” "Servicio nacional de la discapacidad". [Online]. Available: www.senadis.gob.cl/descarga/i/2285/documento

[21] ——, “Manual de procedimientos de ayudas técnicas,” "Servicio nacional de la discapacidad". [Online]. Available: www.senadis.gob.cl/descarga/i/223/documento

[22] J. Martin, *Rapid application development*. Macmillan Publishing Co., Inc., 1991.

[23] K. Pernice and J. Nielsen, “Beyond alt text: Making the web easy to use for users with disabilities,” *California, USA: Nielsen Norman Group*, 2002.

[24] I. Sommerville, *Ingeniería del software*. Pearson Educación, 2005.

A. APÉNDICE

A.1 CÓDIGO DE MODELOS EXPERIMENTALES

A.1.1 Entrenamiento de Red Neuronal “Grande”

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Activation
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
matplotlib.use('agg')
import matplotlib.pyplot as plt

# Fixed seed for experiment replayability
seed = 2141
np.random.seed(seed)

filenames = 'big'
n_classes = 10

# Dataset download (If not local, from Internet)
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Building the input vector from the 28x28 pixels
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

# Normalizing the data
X_train /= 255
X_test /= 255

# One-hot encoding
Y_train = np_utils.to_categorical(y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)

# Model design
def larger_model():
    # create model
    model = Sequential()
    model.add(Conv2D(30, (5, 5), input_shape=(1, 28, 28), activation='relu',
        data_format='channels_first'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(15, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(50, activation='relu'))
```

```
    model.add(Dense(n_classes, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['
        accuracy'])
    return model

# Model building
model = larger_model()

# Model compiling
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=
    'adam')

# Model training and metrics capture
history = model.fit(X_train, Y_train,
          batch_size=100, epochs=20,
          verbose=1,
          validation_data=(X_test, Y_test))

# Saving model
model.save(filenames + '_model.h5')
# Saving weights
model.save_weights(filenames + '_tensors.h5')
print('Model saved.')

# Plotting the metrics
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.tight_layout()

plt.savefig(filenames + '_build_met.png')
print("Metrics saved as " + filenames + "'_build_met.png'.")

loss_and_metrics = model.evaluate(X_test, Y_test, verbose=2)

print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])

# load the model and create predictions on the test set
predicted_classes = model.predict_classes(X_test)

# See which we predicted correctly and which not
correct_indices = np.nonzero(predicted_classes == y_test)[0]
incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
```

```
print()
print(len(correct_indices)," classified correctly")
print(len(incorrect_indices)," classified incorrectly")

# Adapt figure size to accomodate 18 subplots
plt.rcParams['figure.figsize'] = (7,14)

figure_evaluation = plt.figure()

# Plot 9 correct predictions
for i, correct in enumerate(correct_indices[:9]):
    plt.subplot(6,3,i+1)
    plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none
        ')
    plt.title(
      "Predicted: {}, Truth: {}".format(predicted_classes[correct],
                                        y_test[correct]))
    plt.xticks([])
    plt.yticks([])

# Plot 9 incorrect predictions
for i, incorrect in enumerate(incorrect_indices[:9]):
    plt.subplot(6,3,i+10)
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='
        none')
    plt.title(
      "Predicted {}, Truth: {}".format(predicted_classes[incorrect],
                                       y_test[incorrect]))
    plt.xticks([])
    plt.yticks([])

figure_evaluation

plt.savefig(filenames + '_build_pred.png')
print("Predictions saved as '" + filenames + "_build_pred.png'.")
```

A.1.2 Clasificación con Red Neuronal “Grande”

```
import numpy as np
from keras.datasets import mnist
from keras.models import load_model
from keras.utils import np_utils
from datetime import datetime
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import zmq, sys, pickle, argparse, copy, math

filenames = 'big'
port = '5000'
port_end = '5001'

# Fijar semilla para reproducir experimento
seed = 2141
```

```
np.random.seed(seed)

# Descargar dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

## Gets IP and PORT from command line and parses them
ConnectionInfo = argparse.ArgumentParser()
ConnectionInfo.add_argument("-i",  default='127.0.0.1')
ConnectionInfo.add_argument("-o",  default='0.0.0.0')
ConnectionInfo.add_argument("-c",  default=math.floor(np.random.random() *
    1000))
ConnectionInfoParsed = ConnectionInfo.parse_args()

# Saves the parsed IP and Port
ip_in = ConnectionInfoParsed.i
ip_out = ConnectionInfoParsed.o
sample = int(ConnectionInfoParsed.c)

print("MNIST Sample", int(sample))
foo = X_test[sample]
print("Expected class: ", y_test[sample])

#Inicia medicion de tiempo
start = datetime.now()

print('Image preprocessing...')
foo = np.expand_dims(foo, axis=0)
message = foo

# building the input vector from the 28x28 pixels
foo = foo.reshape(foo.shape[0], 1, 28, 28).astype('float32')
#X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

# normalizing the data to help with the training
foo /= 255

print('Preprocessing done.')
print('Loading model and tensors...')

# Cargar modelo preguardado
model = load_model(filenames + '_model.h5')
model.load_weights(filenames + '_tensors.h5')

print('Model loading done.')
print('Classifying...')

# load the model and create predictions on the test set
class_start = datetime.now()
predicted_classes = model.predict_classes(foo)
class_end = datetime.now() - class_start

print('-----------------------------')
print("Value predicted: ", predicted_classes)
print('Classification done in (hh:mm:ss.ms) {}'.format(class_end))
print("Predicted class: ", predicted_classes)

total = datetime.now() - start
print('Network processing time (hh:mm:ss.ms) {}'.format(total))
print('Done!')
```

A.1.3 Entrenamiento de Red Neuronal “Pequeña”

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Activation
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt

# Fixed seed for experiment replayability
seed = 2141
np.random.seed(seed)

filenames = '01_u'
n_classes = 3

# Dataset download (If not local, from Internet)
(X_train, y_train), (X_test, y_test) = mnist.load_data()

###############################
# Re-tagging the dataset for the specific classes we want to train
# 0 -> Class 1
# 1 -> Class 2
# 2 -> 'Others'
y_train[y_train==0]=0
y_train[y_train==1]=1
y_train[y_train>=2]=2

y_test[y_test==0]=0
y_test[y_test==1]=1
y_test[y_test>=2]=2
###############################

# Building the input vector from the 28x28 pixels
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

# Normalizing the data
X_train /= 255
X_test /= 255

# One-hot encoding
Y_train = np_utils.to_categorical(y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)

# Model design
def baseline_model():
    model = Sequential()
```

```
    model.add(Conv2D(32, (5, 5), input_shape=(1, 28, 28), activation='relu',
        data_format='channels_first'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(n_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['
        accuracy'])
    return model

# Model building
model = baseline_model()

# Model compiling
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=
    'adam')

# Model training and metrics capture
history = model.fit(X_train, Y_train,
          batch_size=200, epochs=5,
          verbose=1,
          validation_data=(X_test, Y_test))

# Saving model
model.save(filenames + '_model.h5')
# Saving weights
model.save_weights(filenames + '_tensors.h5')
print('Model saved.')

# Plotting the metrics
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.tight_layout()

plt.savefig(filenames + '_build_met.png')
print("Metrics saved as " + filenames + "'_build_met.png'.")

loss_and_metrics = model.evaluate(X_test, Y_test, verbose=2)

print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])

# load the model and create predictions on the test set
```

```
predicted_classes = model.predict_classes(X_test)

# See which we predicted correctly and which not
correct_indices = np.nonzero(predicted_classes == y_test)[0]
incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
print()
print(len(correct_indices)," classified correctly")
print(len(incorrect_indices)," classified incorrectly")

# Adapt figure size to accomodate 18 subplots
plt.rcParams['figure.figsize'] = (7,14)

figure_evaluation = plt.figure()

# Plot 9 correct predictions
for i, correct in enumerate(correct_indices[:9]):
    plt.subplot(6,3,i+1)
    plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none
        ')
    plt.title(
      "Predicted: {}, Truth: {}".format(predicted_classes[correct],
                                        y_test[correct]))
    plt.xticks([])
    plt.yticks([])

# Plot 9 incorrect predictions
for i, incorrect in enumerate(incorrect_indices[:9]):
    plt.subplot(6,3,i+10)
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='
        none')
    plt.title(
      "Predicted {}, Truth: {}".format(predicted_classes[incorrect],
                                       y_test[incorrect]))
    plt.xticks([])
    plt.yticks([])

figure_evaluation

plt.savefig(filenames + '_build_pred.png')
print("Predictions saved as '" + filenames + "_build_pred.png'.")
```

A.1.4 Clasificación con Red Neuronal “Pequeña”

```
import numpy as np
from keras.datasets import mnist
from keras.models import load_model
from keras.utils import np_utils
from datetime import datetime
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import zmq, sys, pickle, argparse, copy, math

filenames = '01_u'
```

```
port = '5000'
port_end = '5001'

# Fijar semilla para reproducir experimento
seed = 2141
np.random.seed(seed)

# Descargar dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

## Gets IP and PORT from command line and parses them
ConnectionInfo = argparse.ArgumentParser()
ConnectionInfo.add_argument("-i",  default='127.0.0.1')
ConnectionInfo.add_argument("-o",  default='0.0.0.0')
ConnectionInfo.add_argument("-c",  default=math.floor(np.random.random() *
    1000))
ConnectionInfoParsed = ConnectionInfo.parse_args()

# Saves the parsed IP and Port
ip_in = ConnectionInfoParsed.i
ip_out = ConnectionInfoParsed.o
sample = int(ConnectionInfoParsed.c)

print("MNIST Sample", int(sample))
foo = X_test[sample]
print("Expected class: ", y_test[sample])

#Inicia medicion de tiempo
start = datetime.now()

print('Image preprocessing...')
foo = np.expand_dims(foo, axis=0)
message = foo

# building the input vector from the 28x28 pixels
foo = foo.reshape(foo.shape[0], 1, 28, 28).astype('float32')
#X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

# normalizing the data to help with the training
foo /= 255

print('Preprocessing done.')
print('Loading model and tensors...')

# Cargar modelo preguardado
model = load_model(filenames + '_model.h5')
model.load_weights(filenames + '_tensors.h5')

print('Model loading done.')
print('Classifying...')

# load the model and create predictions on the test set
class_start = datetime.now()
predicted_classes = model.predict_classes(foo)
class_end = datetime.now() - class_start

print('-------------------------------')
print("Value predicted: ", predicted_classes)
print('Classification done in (hh:mm:ss.ms) {}'.format(class_end))
```

```
if predicted_classes == 2:
    print("Predicted class: 'other'...")
    print("Continuing classification at next node...")
    # ZeroMQ Context
    context = zmq.Context()
    # Preparing ZeroMQ context for the next node...
    sock = context.socket(zmq.REQ)
    sock.bind('tcp://0.0.0.0:'+port)
    sock.send(pickle.dumps(message))
    X_answer = sock.recv()
    personal = datetime.now() - start
    print('Node processing time (hh:mm:ss.ms) {}'.format(personal))
    print('Data sent. Waiting for classification...')
    sock.close()

    # Espera hasta que concluya la clasificacion
    sock = context.socket(zmq.REQ)
    sock.bind('tcp://0.0.0.0:'+port_end)
    for x in range (1, 6):
        sock.send_string('ack')
        result = sock.recv()
        result = pickle.loads(result)
        if result == -1:
            print("Node "+ str(x) +" couldn't classify your sample.")
        else:
            print("Node "+ str(x) +" predicted class: ", result)
            break
    sock.close()
else:
    print("Predicted class: ", predicted_classes)

total = datetime.now() - start
print('Network processing time (hh:mm:ss.ms) {}'.format(total))
print('Done!')
```

A.2 CÓDIGO PARA ARQUITECTURA RED DE REDES NEURONALES

A.2.1 Entrenamiento de Red Neuronal “Nodo 1”

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Activation
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import matplotlib
matplotlib.use('agg')
```

```
14 import matplotlib.pyplot as plt
15
16 # Fixed seed for experiment replayability
17 seed = 2141
18 np.random.seed(seed)
19
20 filenames = '01_u'
21 n_classes = 3
22
23 # Dataset download (If not local, from Internet)
24 (X_train, y_train), (X_test, y_test) = mnist.load_data()
25
26 ################################
27 # Re-tagging the dataset for the specific classes we want to train
28 # 0 -> Class 1
29 # 1 -> Class 2
30 # 2 -> 'Others'
31 y_train[y_train==0]=0
32 y_train[y_train==1]=1
33 y_train[y_train>=2]=2
34
35 y_test[y_test==0]=0
36 y_test[y_test==1]=1
37 y_test[y_test>=2]=2
38 ################################
39
40 # Building the input vector from the 28x28 pixels
41 X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
42 X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
43
44 # Normalizing the data
45 X_train /= 255
46 X_test /= 255
47
48 # One-hot encoding
49 Y_train = np_utils.to_categorical(y_train, n_classes)
50 Y_test = np_utils.to_categorical(y_test, n_classes)
51
52 # Model design
53 def baseline_model():
54     model = Sequential()
55     model.add(Conv2D(32, (5, 5), input_shape=(1, 28, 28), activation='relu',
          data_format='channels_first'))
56     model.add(MaxPooling2D(pool_size=(2, 2)))
57     model.add(Dropout(0.2))
58     model.add(Flatten())
59     model.add(Dense(128, activation='relu'))
60     model.add(Dense(n_classes, activation='softmax'))
61     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['
          accuracy'])
62     return model
63
64 # Model building
65 model = baseline_model()
66
67 # Model compiling
68 model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=
      'adam')
69
```

```
# Model training and metrics capture
history = model.fit(X_train, Y_train,
          batch_size=200, epochs=5,
          verbose=1,
          validation_data=(X_test, Y_test))

# Saving model
model.save(filenames + '_model.h5')
# Saving weights
model.save_weights(filenames + '_tensors.h5')
print('Model saved.')

# Plotting the metrics
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.tight_layout()

plt.savefig(filenames + '_build_met.png')
print("Metrics saved as " + filenames + "'_build_met.png'.")

loss_and_metrics = model.evaluate(X_test, Y_test, verbose=2)

print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])

# load the model and create predictions on the test set
predicted_classes = model.predict_classes(X_test)

# See which we predicted correctly and which not
correct_indices = np.nonzero(predicted_classes == y_test)[0]
incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
print()
print(len(correct_indices)," classified correctly")
print(len(incorrect_indices)," classified incorrectly")

# Adapt figure size to accomodate 18 subplots
plt.rcParams['figure.figsize'] = (7,14)

figure_evaluation = plt.figure()

# Plot 9 correct predictions
for i, correct in enumerate(correct_indices[:9]):
    plt.subplot(6,3,i+1)
```

```
    plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none
        ')
    plt.title(
      "Predicted: {}, Truth: {}".format(predicted_classes[correct],
                                        y_test[correct]))
    plt.xticks([])
    plt.yticks([])

# Plot 9 incorrect predictions
for i, incorrect in enumerate(incorrect_indices[:9]):
    plt.subplot(6,3,i+10)
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='
        none')
    plt.title(
      "Predicted {}, Truth: {}".format(predicted_classes[incorrect],
                                       y_test[incorrect]))
    plt.xticks([])
    plt.yticks([])

figure_evaluation

plt.savefig(filenames + '_build_pred.png')
print("Predictions saved as '" + filenames + "_build_pred.png'.")
```

A.2.2 Clasificación con Red Neuronal “Nodo 1”

```
import numpy as np
from keras.datasets import mnist
from keras.models import load_model
from keras.utils import np_utils
from datetime import datetime
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import zmq, sys, pickle, argparse, copy, math

filenames = '01_u'
port = '5000'
port_end = '5001'

# Fijar semilla para reproducir experimento
seed = 2141
np.random.seed(seed)

# Descargar dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

## Gets IP and PORT from command line and parses them
ConnectionInfo = argparse.ArgumentParser()
ConnectionInfo.add_argument("-i", default='127.0.0.1')
ConnectionInfo.add_argument("-o", default='0.0.0.0')
ConnectionInfo.add_argument("-c", default=math.floor(np.random.random() *
    1000))
ConnectionInfoParsed = ConnectionInfo.parse_args()
```

```
28
29  # Saves the parsed IP and Port
30  ip_in = ConnectionInfoParsed.i
31  ip_out = ConnectionInfoParsed.o
32  sample = int(ConnectionInfoParsed.c)
33
34  print("MNIST Sample", int(sample))
35  foo = X_test[sample]
36  print("Expected class: ", y_test[sample])
37
38  #Inicia medicion de tiempo
39  start = datetime.now()
40
41  print('Image preprocessing...')
42  foo = np.expand_dims(foo, axis=0)
43  message = foo
44
45  # building the input vector from the 28x28 pixels
46  foo = foo.reshape(foo.shape[0], 1, 28, 28).astype('float32')
47  #X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
48
49  # normalizing the data to help with the training
50  foo /= 255
51
52  print('Preprocessing done.')
53  print('Loading model and tensors...')
54
55  # Cargar modelo preguardado
56  model = load_model(filenames + '_model.h5')
57  model.load_weights(filenames + '_tensors.h5')
58
59  print('Model loading done.')
60  print('Classifying...')
61
62  # load the model and create predictions on the test set
63  class_start = datetime.now()
64  predicted_classes = model.predict_classes(foo)
65  class_end = datetime.now() - class_start
66
67  print('-----------------------------')
68  print("Value predicted: ", predicted_classes)
69  print('Classification done in (hh:mm:ss.ms) {}'.format(class_end))
70  if predicted_classes == 2:
71      print("Predicted class: 'other'...")
72      print("Continuing classification at next node...")
73      # ZeroMQ Context
74      context = zmq.Context()
75      # Preparing ZeroMQ context for the next node...
76      sock = context.socket(zmq.REQ)
77      sock.bind('tcp://0.0.0.0:'+port)
78      sock.send(pickle.dumps(message))
79      X_answer = sock.recv()
80      personal = datetime.now() - start
81      print('Node processing time (hh:mm:ss.ms) {}'.format(personal))
82      print('Data sent. Waiting for classification...')
83      sock.close()
84
85      # Espera hasta que concluya la clasificacion
86      sock = context.socket(zmq.REQ)
```

```
    sock.bind('tcp://0.0.0.0:'+port_end)
    for x in range (1, 6):
        sock.send_string('ack')
        result = sock.recv()
        result = pickle.loads(result)
        if result == -1:
            print("Node "+ str(x) +" couldn't classify your sample.")
        else:
            print("Node "+ str(x) +" predicted class: ", result)
            break
    sock.close()
else:
    print("Predicted class: ", predicted_classes)

total = datetime.now() - start
print('Network processing time (hh:mm:ss.ms) {}'.format(total))
print('Done!')
```

A.2.3 Clasificación con Red Neuronal "Nodo 2"

```
import numpy as np
from keras.models import load_model
from keras.utils import np_utils
from datetime import datetime
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import zmq, pickle, sys, argparse, copy

filenames = '23_u'
port = '5000'
port_out = '5001'

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.REP)

## Gets IP and PORT from command line and parses them
ConnectionInfo = argparse.ArgumentParser()
ConnectionInfo.add_argument("-i",  default='127.0.0.1')
ConnectionInfo.add_argument("-o",  default='0.0.0.0')
ConnectionInfoParsed = ConnectionInfo.parse_args()

# Saves the parsed IP and Port
ip_in = ConnectionInfoParsed.i
ip_out = ConnectionInfoParsed.o

try:
    sock.connect('tcp://'+ip_in+':'+port)
except:
    print('Usage: python load_'+filenames+'.py -i <valid input ip address> -o <
        valid output ip address>')
```

```
34
35  # Fijar semilla para reproducir experimento
36  seed = 2141
37  np.random.seed(seed)
38
39  # Run a simple "Echo" server
40  print('Listening to tcp://'+ip_in+':'+port+'...')
41  X_message = sock.recv()
42  print('Receiving data...')
43  X_test = pickle.loads(X_message)
44  sock.send(pickle.dumps(X_message))
45  sock.close()
46  print('Data received. Starting classification...')
47  message = X_test
48
49  global_start = datetime.now()
50
51  # building the input vector from the 28x28 pixels
52  X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
53
54  # normalizing the data to help with the training
55  X_test /= 255
56
57  # Cargar modelo preguardado
58  model = load_model(filenames + '_model.h5')
59  model.load_weights(filenames + '_tensors.h5')
60
61  # load the model and create predictions on the test set
62  class_start = datetime.now()
63  predicted_classes = model.predict_classes(X_test)
64  class_end = datetime.now() - class_start
65
66
67  print('_________________________________')
68  print("Value predicted: ", predicted_classes)
69  print('Node classification done in (hh:mm:ss.ms) {}'.format(class_end))
70  if predicted_classes == 2:
71      print("Predicted class: 'other'...")
72      print("Continuing classification at next node...")
73      # ZeroMQ Context
74      context = zmq.Context()
75      # Preparing ZeroMQ context for the next node...
76      sock = context.socket(zmq.REQ)
77      sock.bind('tcp://0.0.0.0:'+port)
78      sock.send(pickle.dumps(message))
79      X_answer = sock.recv()
80      print('Data sent to next node.')
81      sock.close()
82  else:
83      print("predicted class: ", predicted_classes)
84      # ZeroMQ Context
85      context = zmq.Context()
86      sock = context.socket(zmq.REP)
87      sock.connect('tcp://'+ip_out+':'+port_out)
88      end_string = sock.recv()
89      sock.send(pickle.dumps(predicted_classes+2))
90
91  global_end = datetime.now() - global_start
92
```

```
print('Node processing done in (hh:mm:ss.ms) {}'.format(global_end))
print('Done!')
```

A.2.4 Clasificación con Red Neuronal “Nodo n”

```
import numpy as np
from keras.models import load_model
from keras.utils import np_utils
from datetime import datetime
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import zmq, pickle, sys, argparse, copy

filenames = '89_u'
port = '5000'
port_out = '5001'

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.REP)

## Gets IP and PORT from command line and parses them
ConnectionInfo = argparse.ArgumentParser()
ConnectionInfo.add_argument("-i", default='127.0.0.1')
ConnectionInfo.add_argument("-o", default='0.0.0.0')
ConnectionInfoParsed = ConnectionInfo.parse_args()

# Saves the parsed IP and Port
ip_in = ConnectionInfoParsed.i
ip_out = ConnectionInfoParsed.o

try:
    sock.connect('tcp://'+ip_in+':'+port)
except:
    print('Usage: python load_'+filenames+'.py -i <valid input ip address> -o <
        valid output ip address>')


# Fijar semilla para reproducir experimento
seed = 2141
np.random.seed(seed)


# Run a simple "Echo" server
print('Listening to tcp://'+ip_in+':'+port+'...')
X_message = sock.recv()
print('Receiving data...')
X_test = pickle.loads(X_message)
sock.send(pickle.dumps(X_message))
sock.close()
print('Data received. Starting classification...')
```

```
49
50  global_start = datetime.now()
51
52
53  # building the input vector from the 28x28 pixels
54  X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
55
56
57  # normalizing the data to help with the training
58  X_test /= 255
59
60
61  # Cargar modelo preguardado
62  model = load_model(filenames + '_model.h5')
63  model.load_weights(filenames + '_tensors.h5')
64
65  # load the model and create predictions on the test set
66  class_start = datetime.now()
67  predicted_classes = model.predict_classes(X_test)
68  class_end = datetime.now() - class_start
69
70  print('-------------------------------')
71  print("Value predicted: ", predicted_classes)
72  print('Node classification done in (hh:mm:ss.ms) {}'.format(class_end))
73
74  if predicted_classes == 2:
75      print("Predicted class: 'other'...")
76      print("Network couldn't find a class for the sample data. Returning to
            first node.")
77      # ZeroMQ Context
78      context = zmq.Context()
79      # Preparing ZeroMQ context for the next node...
80      sock = context.socket(zmq.REQ)
81      sock.connect('tcp://'+ip_out+':'+port_out)
82      sock.send(pickle.dumps(-1))
83      X_answer = sock.recv()
84      sock.close()
85
86  else:
87      print("predicted class: ", predicted_classes)
88
89      # ZeroMQ Context
90      context = zmq.Context()
91      sock = context.socket(zmq.REP)
92      sock.connect('tcp://'+ip_out+':'+port_out)
93      end_string = sock.recv()
94      sock.send(pickle.dumps(predicted_classes+8))
95      sock.close()
96
97  global_end = datetime.now() - global_start
98
99  print('Node processing done in (hh:mm:ss.ms) {}'.format(global_end))
100 print('Done!')
```

A.3 MATRICES DE CONFUSIÓN DE REDES NEURONALES ENTRENADAS

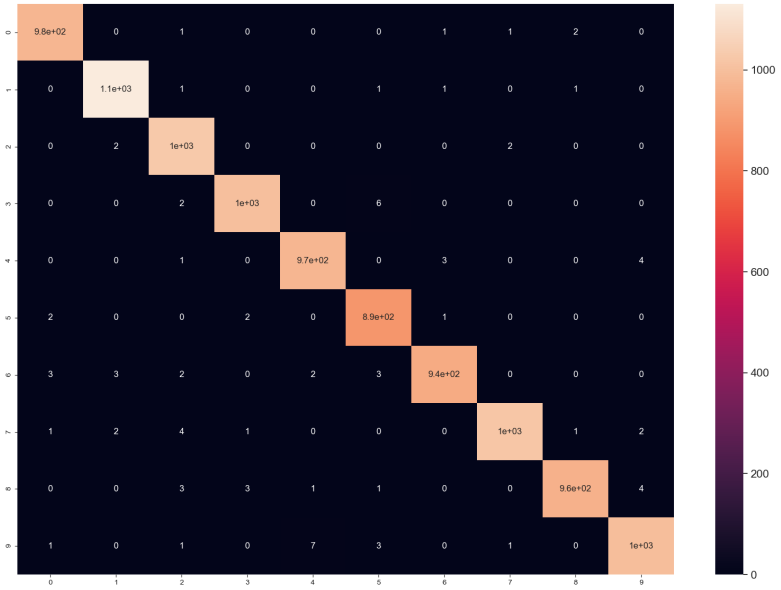

Figura A.1: Matriz de Confusión de RNG - Modelo *Big-Little*. Fuente: Elaboración propia, 2018.

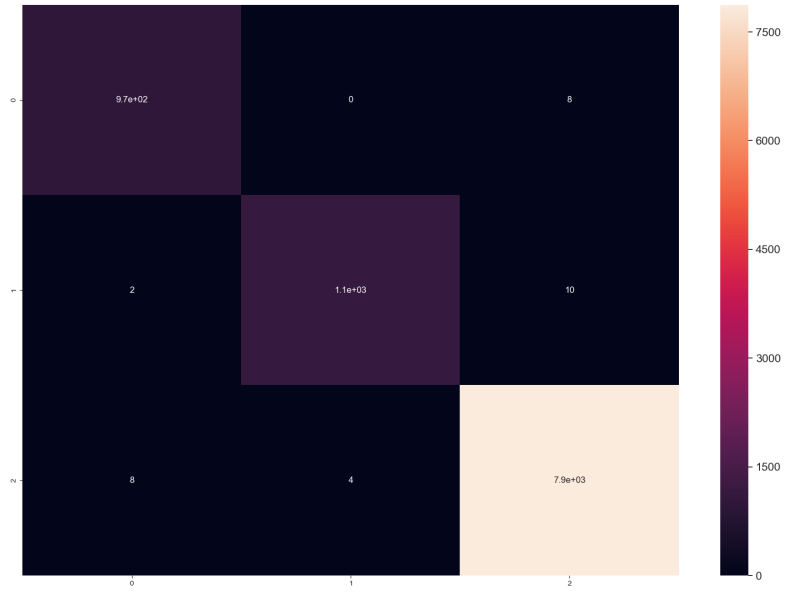

Figura A.2: Matriz de Confusión de RNP - Modelo *Big-Little*. Fuente: Elaboración propia, 2018.

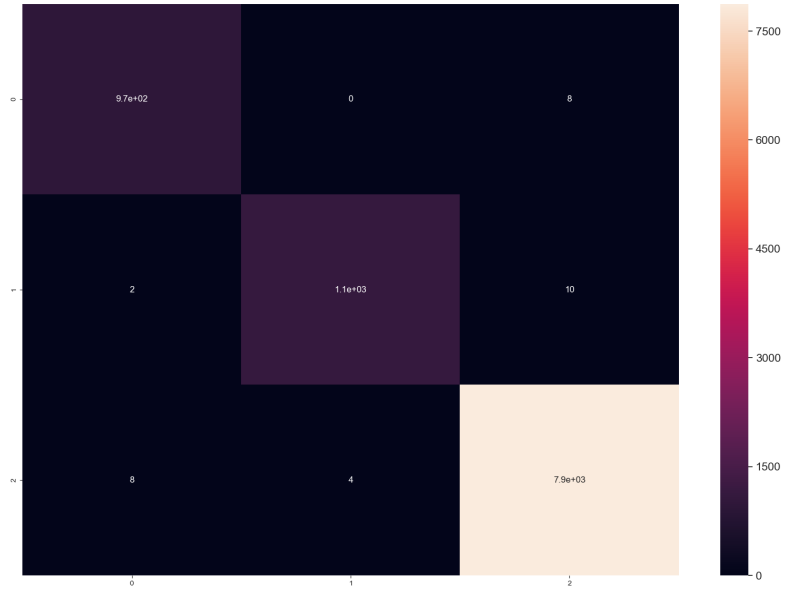

Figura A.3: Matriz de Confusión de Nodo 1 - Modelo RRN. Fuente: Elaboración propia, 2018.

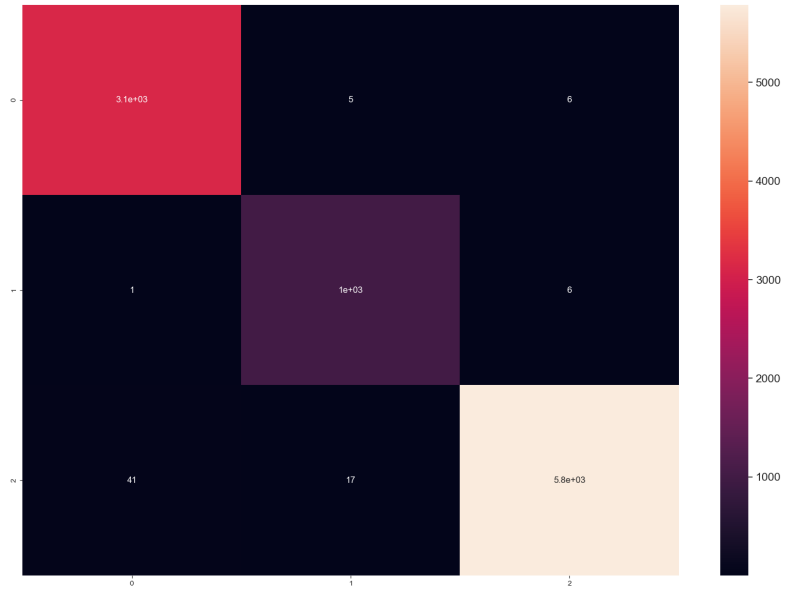

Figura A.4: Matriz de Confusión de Nodo 2 - Modelo RRN. Fuente: Elaboración propia, 2018.

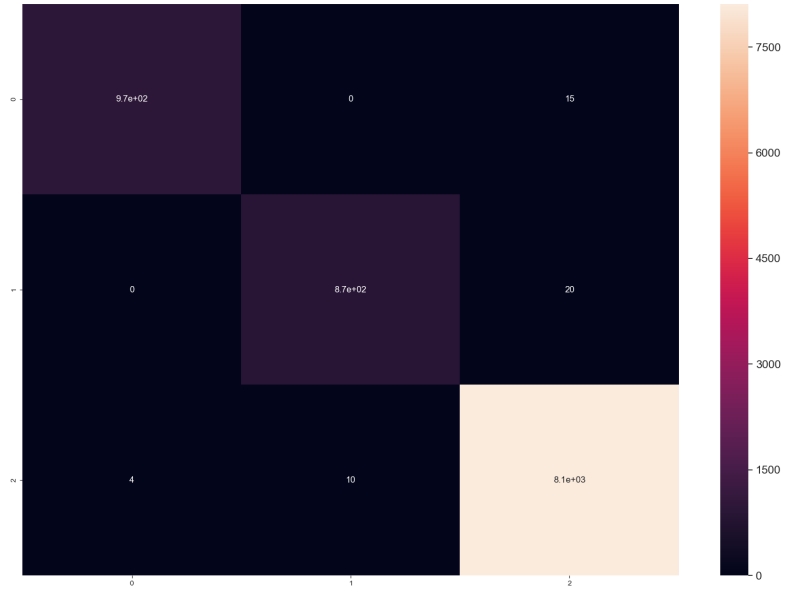

Figura A.5: Matriz de Confusión de Nodo 3 - Modelo RRN. Fuente: Elaboración propia, 2018.

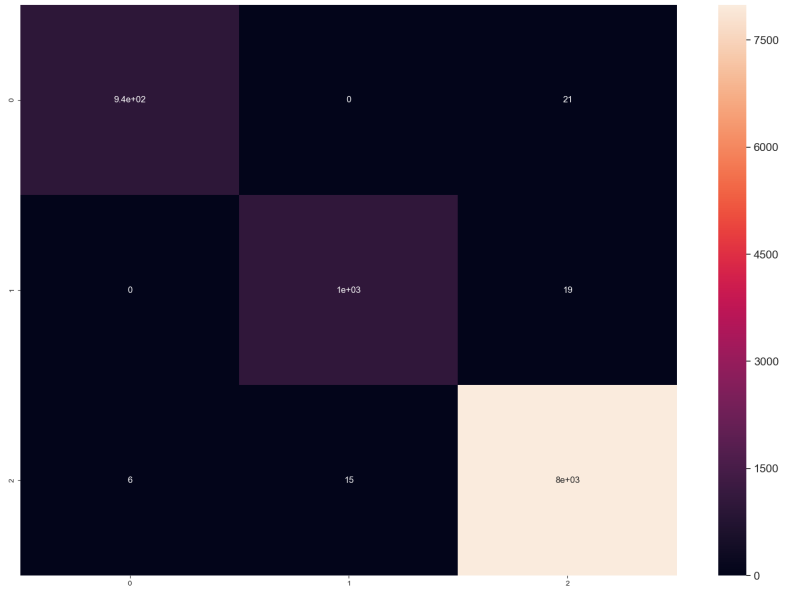

Figura A.6: Matriz de Confusión de Nodo 4 - Modelo RRN. Fuente: Elaboración propia, 2018.

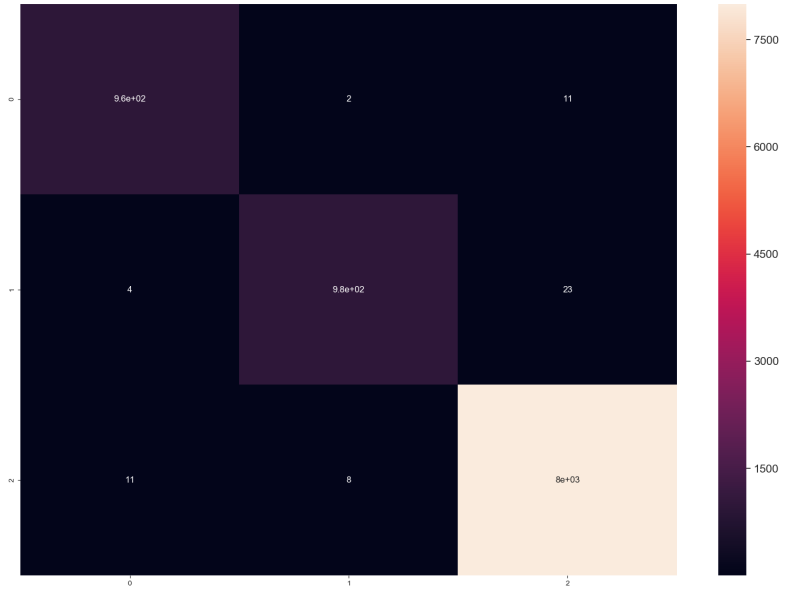

Figura A.7: Matriz de Confusión de Nodo 5 - Modelo RRN. Fuente: Elaboración propia, 2018.