UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Herramienta para la captura y análisis de interacciones en el Entorno de Desarrollo Integrado NetBeans

ROBERTO ANTONIO URETA MUÑOZ

Profesor Guía: LUIS GREGORIO SILVESTRE QUIROGA

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
Julio, 2020

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2022

Dedicado a mis padres, Sandra y Omar...

AGRADECIMIENTOS

A mis padres, Sandra y Omar por su infinito amor e incondicional apoyo durante todos estos años, por sus continuas enseñanzas para afrontar la vida, por los principios y valores inculcados.

A mis hermanos, Joaquín y Carolina, no seria lo mismo sin ustedes. Gracias por su cariño, amor y risas. Espero que sean grandes personas.

A mi Familia, por su unión y cariño entregado. Con especial mención para mis abuelos, Ecinesio y Rudecinda por su gran amor y por ser un pilar fundamental en gran parte de mi vida y para mi tío, Juan Carlos Muñoz, por su especial apoyo y por sus numerosas enseñanzas.

A mis amigos y compañeros, gracias por su constante apoyo para sobrellevar cursos o trabajos, pero, por sobretodo gracias por las incontables risas necesarias para afrontar toda la vida universitaria. Con mención especial para Ariel Cornejo, Diego Iturriaga, Diego Matus, Felipe Milla, José Núñez, Benjamín Pino, Gabriel Sanhueza y Maximiliano Viveros.

A los profesores, por su dedicación entregada en los distintos cursos y su disponibilidad ante cualquier duda. Gracias en especial, a mi profesor guía Luis Silvestre por sus consejos y orientación durante el desarrollo del proyecto.

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

RESUMEN

A día de hoy, muchas empresas relacionadas con la tecnología tienen la necesidad de medir la productividad de sus empleados, con la finalidad de mejorar su rendimiento en la industria. En el ámbito académico, también se presenta este tipo de necesidad pero enfocada al desarrollo técnico de los estudiantes. Sobre todo en carreras como Ingeniería Civil en Computación o Informática.

Concretamente, en la carrera de Ingeniería Civil en Computación de la Universidad de Talca, existen cursos elementales de programación orientados a desarrollar el paradigma de la programación orientada a objetos mediante el trabajo individual o por equipos. En estos cursos por preferencia de los estudiantes se suele utilizar el entorno de desarrollo NetBeans.

Sin embargo, los profesores de dichos cursos elementales de programación carecen de evidencia que sea de utilidad para una evaluación cuantitativa en cuanto al desempeño de los estudiantes. Las estrategias actuales para medir el rendimiento técnico de los estudiantes suelen ser difíciles de aplicar, ya que, conllevan una gran inversión de tiempo para los profesores.

Con el objetivo de abordar la problemática de evaluación cuantitativa de dichos cursos, se construye un módulo/plugin NetBeans llamado *BlueLogger*. Este es capaz de capturar las interacciones de estudiantes a medida que avanzan en los distintos proyectos de programación. Además, se utiliza una aplicación web para que los profesores visualicen a través de métricas la información recopilada en las distintas sesiones de desarrollo de los estudiantes.

La metodología escogida para el desarrollo de BlueLogger es *Personal Extreme Programming*, debido a su diseño orientado a proyectos individuales. Por otro lado, para la evaluación del proyecto se utiliza Experimentación en Ingeniería de Software, la cual delimita los objetivos y características a evaluar.

Los resultados obtenidos en la evaluación del proyecto indican que la aplicación de BlueLogger es factible en cursos de programación elemental. Por otra parte, los profesores pueden visualizar las métricas capturadas usando una aplicación web. Finalmente, se propone como trabajo futuro la integración del módulo con otros editores de código, agregar nuevas métricas o realizar un análisis más complejo con las interacciones capturadas.

1. Introducción

En este capítulo, se describe la situación actual del problema a trabajar centrándose en la contextualización de este y en las justificaciones que se tienen para desarrollar una solución. A continuación, se presenta el contexto, la problemática, los objetivos, la propuesta de solución, los alcances del proyecto y por último, el trabajo relacionado.

1.1. Contexto

En la Facultad de Ingeniería de la Universidad de Talca, específicamente en la carrera de Ingeniería Civil en Computación, existen cursos orientados a la programación individual y/o por equipos. Por ejemplo, se encuentra Programación avanzada y Proyecto de programación los cuales son cursos impartidos en el tercer y cuarto semestre de la carrera, respectivamente.

Los tipos de proyectos abordados buscan que los estudiantes se familiaricen con el paradigma de la programación orientada a objetos y también que conozcan cómo es trabajar en equipo. Usualmente, se aplica el lenguaje Java utilizando entornos de desarrollos como Eclipse, IntelliJ IDEA o NetBeans siendo este último el preferido por la mayoría de estudiantes.

En estos módulos se suele aplicar la metodología de enseñanza basada en competencias [2] que consiste en realizar un seguimiento a lo largo de todo el proceso, que permita obtener información acerca de cómo se está llevando a cabo, con la finalidad de reajustar la intervención orientadora, de acuerdo con los datos recopilados. El uso de esta metodología permite efectuar una constante retroalimentación al curso siendo de gran utilidad para estudiantes y profesores.

1.2. Problema

Actualmente el profesor que dicta los cursos en primeros años asigna una calificación subjetiva sin seguir algún tipo de parámetro o métrica. En otras palabras, el profesor suele asignar una nota basándose en su percepción en cuanto al trabajo realizado por los alumnos. En este sentido, es difícil contar con evidencia que sirva de apoyo para una evaluación cuantitativa según el desempeño individual de los alumnos debido a que el procedimiento para obtener evidencia del avance es tedioso lo que conlleva a una inversión de tiempo considerable. A modo de ilustración, el curso de Programación Avanzada perteneciente al tercer semestre de Ingeniería Civil en Computación carece de métodos para utilizar calificaciones basadas en una evaluación cuantitativa.

De lo anterior, surge un inconveniente a la hora de identificar retrasos, errores o procedimientos sistemáticos quedando a criterio de cada profesor las decisiones tomadas. Junto con esto, surgen impedimentos si el profesor quisiera por ejemplo; comparar las horas autónomas mencionadas en el syllabus con las horas que realmente emplean los estudiantes para programar, otorgar calificaciones basadas en parámetros de relevancia que se pierden durante el desarrollo de proyectos o revisar la factibilidad técnica de trabajos solicitados que debido a algún problema pocos estudiantes hayan entregado lo esperado.

1.3. Objetivos

A continuación, se define el objetivo principal del proyecto seguido de sus objetivos específicos.

1.3.1. Objetivo General

- Desarrollar una herramienta que permita capturar las interacciones de un usuario durante la programación de un proyecto en el IDE NetBeans.

1.3.2. Objetivos Específicos

- Caracterizar las interacciones genéricas que son aplicables en el desarrollo de un proyecto de software.

- Seleccionar según las limitantes del Entorno de Desarrollo Integrado NetBeans las interacciones específicas a capturar.
- Desarrollar un módulo para el Entorno de Desarrollo Integrado NetBeans que sea capaz de capturar las interacciones de usuarios presentes en archivos de código fuente con extensión Java.
- Definir las métricas de desarrollo específicas a utilizar en base a las interacciones capturadas.
- Desarrollar una aplicación web que permita visualizar la información capturada por el módulo a través de métricas de desarrollo.

1.4. Propuesta de Solución

La propuesta consiste en un módulo [1] para el Entorno de Desarrollo Integrado NetBeans que sea capaz de capturar las interacciones producidas por un usuario en las distintas sesiones efectuadas en la programación de un proyecto. La idea es registrar parámetros que no son capturados durante el desarrollo de un proyecto. Luego, estos datos son procesados a través de métricas, las cuales serán elegidas en base a estudios anteriores y según el potencial de las funcionalidades ofrecidas por el IDE NetBeans.

La métrica esencial que se espera reconocer es la duración total de las sesiones donde para cada una se incluye el tiempo efectivo de programación, es decir, el tiempo que transcurre mientras el usuario utiliza el teclado de su computador más el tiempo inactivo el cual considera el tiempo que transcurre mientras el usuario tiene abierto el IDE NetBeans y además no utiliza el teclado del computador. Además, se registra la creación y modificación de clases y métodos dentro de un proyecto, este registro considera datos como la cantidad de cambios efectuados y en qué sesión fueron realizados. Por último, con estos datos se consiguen algunas refactorizaciones [2] efectuadas durante una sesión, para ilustrar esto, es posible capturar información de métodos nuevos, modificados o eliminados en los distintos archivos del proyecto. Por

[1] Un módulo es una pieza de software que se relaciona con una aplicación anfitrión para extender sus funcionalidades [3]

[2] La refactorización es el proceso de cambiar un sistema de software de tal manera que no altera el comportamiento exterior del código pero mejora su estructura interna [4]

último, se capturan eventos de inserción y eliminación producidos por el programador al momento de pegar o eliminar una selección de código.

Para visualizar los datos capturados, se dispone de una aplicación web cuyo objetivo es presentar a los usuarios interesados las métricas obtenidas en base a las distintas sesiones de programación de los estudiantes. Para lograr esto, la aplicación permite crear cursos, agregar estudiantes y generar proyectos individuales o grupales. De esta manera, se logra organizar y visualizar los datos recopilados por el módulo dejando a disposición del profesor información relevante que es de ayuda para evaluar cuantitativamente el rendimiento de un estudiante.

1.5. Alcances

A continuación, se definen los alcances para el desarrollo de este proyecto con la finalidad de acotar el trabajo a realizar:

- Este trabajo se limita a ser desarrollado en el marco de la carrera de Ingeniería Civil en Computación de la Universidad de Talca.
- El módulo a desarrollar solo estará disponible para el Entorno de Desarrollo Integrado NetBeans en su versión 8.2.
- El módulo a desarrollar solo estará disponible para proyectos basados en el lenguaje de programación Java.
- La aplicación web se limita a ser un prototipo para visualizar los datos recopilados.
- Para este proyecto no se considera abordar temas de seguridad informática para el módulo de NetBeans.

1.6. Trabajos relacionados

A continuación, se exponen herramientas relacionadas con la propuesta de solución. Se identifica las cualidades que podrían ser un aporte para la solución al problema. Además, se describen las desventajas que no permiten ajustar estas herramientas a lo especificado en el problema. Buscando generar una solución que sea coherente con el problema.

- WakaTime [5], Herramienta que da a conocer el tiempo que lleva programando un usuario, se encuentra disponible para un gran número de editores, los resultados se encuentran disponibles en una página web. Cuenta con distintos planes que van desde el gratuito que tiene funciones demasiado limitadas hasta los planes de pago que agregan un gran número de funcionalidades.

  La cualidad que se destaca de WakaTime es la disponibilidad de una página web para ver lo resultados obtenidos de las sesiones de programación a lo largo del tiempo. Por otra parte, su plan gratuito es limitado por lo que para proyectos que cuentan con equipos de trabajo es necesario pagar para contar con todas las funcionalidades que se ofrecen, como por ejemplo tableros compartidos para equipos de trabajo, exportación de la actividad del usuario y del equipo, además de contar con soporte 24/7.

- Codealike [6], Herramienta que rastrea y analiza la actividad del programador, separa el tiempo de programación en debugging, compilando, escribiendo código o solo leyendo. Las funcionalidades de análisis y las destinadas al trabajo en equipo son limitadas por sus planes premium.

  La ventaja que presenta Codealike es su capacidad para identificar y separar en distintas categorías el tiempo que llevas escribiendo código siendo de utilidad para buscar posibles mejoras en base al tiempo invertido en el desarrollo de un proyecto. Su principal inconveniente al igual que WakaTime es que para acceder a funcionalidades que involucren equipos de trabajo se debe pagar por planes premium.

- Coding Tracker [7], Extensión para VSCode que rastrea el tiempo de codificación y genera informes básicos con esta información. La herramienta no cuenta con una página web para visualizar los reportes, por lo que la información recopilada se encuentra disponible de manera local.

  La principal cualidad de Coding Tracker es que separa el tiempo que llevas escribiendo código del tiempo que llevas inactivo en VSCode además todas sus funcionalidades son gratuitas. Su principal desventaja es que no cuenta con una página web para visualizar los reportes de las sesiones de programación y si se requiere subir esta información a un servidor la configuración debe ser realizada por uno mismo siguiendo ciertos pasos descritos por Coding Tracker.

- Hubstaff [8], Disponible como una aplicación de escritorio, web o móvil que ofrece realizar seguimiento de tiempo, rastreo por GPS y monitoreo de la productividad de los usuarios que la utilicen generando diversos informes en base a la información capturada. Cuenta con una prueba gratuita de 14 días.

  Esta herramienta se destaca por hacer un seguimiento de tiempo del trabajo realizado entregando información muy valiosa a través de su pagina web, el usuario tiene la capacidad de comenzar y detener el seguimiento cuando quiera. Además, si se deja de utilizar el computador el usuario cuenta con un margen de 5 minutos para que el seguimiento no se detenga. Los puntos negativos encontrados guardan relación con la poca privacidad que entrega la herramienta ya que su monitorio de productividad rastrea aplicaciones usadas, urls visitadas y almacenamiento de capturas de pantalla en periodos de tiempo aleatorios.

A continuación, en la Tabla 1.1 se muestra una comparativa para módulos construidos específicamente para entornos de desarrollo que soportan el lenguaje de programación Java:

Tabla 1.1: Características y limitantes de módulos que soportan el lenguaje de programación Java y usan métricas de desarrollo.

Herramienta	Tipo	IDE	Ventajas	Desventajas
CodeMR [9]	Comercial	Eclipse	Soporte para Java, Kotlin, Scala y C++, visualización de métricas de código y atributos de calidad de alto nivel (Acoplamiento, Complejidad, Tamaño) en diferentes vistas.	Orientado a empresas por lo que tiene límites para algunas métricas. Existe un plan de pago para acceder a todo lo que ofrece.
CodeCity [10]	Libre	Eclipse	Visualización intuitiva de las métricas, integración con otras herramientas para mostrar más información.	La versión para eclipse solo muestra el número de métodos declarados, atributos declarados y el de marcadores de problemas.
Metriculator [11]	Libre	Eclipse	Distintas formas de exportación de las métricas, límites configurables para métricas, diseñado para agregar métricas propias.	Solo está disponible para C++ y un limitado número de métricas disponibles.
Trace Compass [12]	Libre	Eclipse	Proporciona vistas, gráficos y métricas. Es posible revisar la información en modo offline.	Orientado a ser utilizado por empresas y entrega demasiada información lo que hace que sea poco intuitivo.
Simple Code Metrics [13]	Libre	NetBeans	Entre las métricas que miden se encuentran la complejidad ciclomática y la falta de cohesión en los métodos.	No existe soporte para clases internas y las variables locales no pueden ocultar las variables de clases. No existen actualizaciones para versiones recientes de NetBeans.
SourceCodeMetric [14]	Libre	NetBeans	Métricas para métodos, clases y paquetes. También permite la generación de reportes.	No existen actualizaciones para versiones recientes.

El principal inconveniente de la mayoría de herramientas expuestas en la Tabla 1.1 es que se encuentran disponibles solo para el entorno de desarrollo Eclipse, IDE que tiene muy bajo uso por parte de estudiantes de primeros años de la carrera

los cuales suelen preferir NetBeans. Lamentablemente, las dos herramientas mencionadas que trabajan usando NetBeans no cuentan con soporte para las versiones recientes de este IDE por lo que actualmente su utilidad es casi nula.

Las diferentes herramientas vistas cumplen con la tarea de capturar métricas que pueden ser de gran utilidad para una persona u organización interesada en por ejemplo conocer el tiempo de programación empleado por un usuario. Además, esta información puede ser presentada en distintos reportes según las necesidades de un desarrollador o de una organización que cuenta con equipos de trabajo.

En cuanto a las limitantes, en el caso de WakaTime y Codealike al ser módulos que funcionan en distintos entornos de desarrollo suelen ser utilizados en un contexto empresarial por ende sus funcionalidades más potentes solo están disponibles para planes de pago, dejando limitadas opciones para ser empleadas en el marco educativo. Algo similar sucede con Hubstaff ya que es una herramienta usada por empresas, es de pago y puede llegar a ser excesivamente invasiva para el usuario ya que constantemente esta tomando capturas de pantalla, es capaz de rastrear paginas web visitadas e incluso tiene la opción de rastreo GPS.

Por otro lado, las herramientas gratuitas disponibles no están actualizadas a versiones recientes de NetBeans o se encuentran habilitadas para otros IDEs como Eclipse o VSCode, entornos de desarrollo que no son parte del contexto planteado.

2. Marco Teórico

En este capítulo se explica la teoría relacionada con las interacciones de usuario presentes en el desarrollo de un proyecto de software seguido de las métricas de software que se pueden obtener usando esta información, también se aborda la arquitectura microkernel usada para implementar el módulo que utiliza NetBeans. Además, se da a conocer las metodologías, herramientas y lenguajes que se usarán para generar la solución a la problemática previamente expuesta otorgando el conocimiento que necesita el lector.

2.1. Interacciones de usuario en el desarrollo de software

Las interacciones presentes en el desarrollo de un proyecto de software tienen que ver con las acciones que toma el programador durante el desarrollo de la solución a un problema. Este camino generalmente contiene algunos errores e inconsistencias, situaciones que se van solucionado a medida que el proyecto avanza [15]. En esta evolución de software, se pierde el proceso de implementación y de cambio, ya que no se registran las sesiones de programación. Lo mas cercano a este tipo de información son los sistemas de versiones manteniendo un modelo superficial de lo que el programador ha hecho [15].

Los cambios compuestos que se suelen apreciar durante el desarrollo de un proyecto se pueden dividir en tipos, los cuáles se explican a continuación:

1. Las **acciones a nivel de desarrollador** corresponden a un cambio unitario desde el punto de vista del programador. Por ejemplo, al cambiar la definición de una clase. Pero en realidad se compone de varios cambios atómicos, para ilustrar, cuando se crea un nuevo paquete se manejan 3 cambios atómicos: la

creación del paquete, la adición de este a la raíz del proyecto y el cambio de nombre del paquete [16].

2. Las **refactorizaciones** son el proceso de cambiar un sistema de software de tal manera que no altera el comportamiento exterior del código pero mejora su estructura interna [4]. Hoy en día, los IDEs automatizan la mayoría de refactorizaciones, por ejemplo, al cambiar el nombre de un método se presenta un renombrado de este por cada referencia presente en la clase o el proyecto.

3. Las **sesiones de desarrollo** agrupan todas las acciones a nivel de desarrollador y refactorizaciones que ocurrieron en una sesión de codificación en el IDE [16].

4. Otros posibles cambios pueden ser la **corrección de errores** o la **implementación de una nueva característica** [16]. Tareas que pueden involucrar varias sesiones, por lo cual, no necesariamente se relacionan con una sola acción a nivel de desarrollo, refactorización o sesión.

2.2. Métricas en el desarrollo de software

Una métrica en el desarrollo de software es una medida que brinda información específica en base al código de una aplicación permitiendo conocer características que pueden ser útiles para ciertos usuarios [17].

Según las interacciones vistas anteriormente, es posible calcular dos tipos de métricas: las métricas a nivel de programa y las métricas a nivel de cambio. Este tipo de métricas son evolutivas, ya que se pueden calcular luego de cada sesión de desarrollo o bien en distintos rangos de tiempo como semanas o meses [16]. En la Tabla 2.1 y 2.2 se muestran ejemplos de métricas genéricas que se pueden obtener en base a las interacciones obtenidas durante el desarrollo de software.

Tabla 2.1: Métricas a nivel de programa.

Métrica	**Descripción**
NDC	Número de clases
NDM	Número de métodos
NDD	Número de declaraciones
TPM	Tamaño promedio de los métodos

Tabla 2.2: Métricas a nivel de cambios.

Métrica	**Descripción**
NEAX	Número de entidades agregadas, las entidades pueden ser paquetes (NEAP), clases (NEAC), métodos (NEAM), etc.
NEMX	Número de entidades modificadas, las entidades pueden ser paquetes (NEMP), clases (NEMC), métodos (NEMM), etc.
NEDX	Número de entidades eliminadas, las entidades pueden ser paquetes (NEDP), clases (NEDC), métodos (NEDM), etc.

La finalidad de las métricas es entregar a un usuario la oportunidad de evaluar, mejorar o clasificar el software final. Las tablas anteriores muestran ejemplos genéricos de métricas. Al centrarse en la obtención de métricas de las distintas sesiones de desarrollo, es posible conseguir indicadores sobre lo que esta ocurriendo en un proyecto de software. En la Tabla 2.3 se destacan las métricas y sus posibles indicadores.

Tabla 2.3: Métricas de una sesión de desarrollo.

Métrica	Descripción	Indicador de
DSM	**Duración** de la **sesión**: expresada en **minutos**.	Cantidad de tiempo total empleado en una sesión.
TAS	**Tiempo activo** durante una **sesión**.	Cantidad de tiempo empleado en escribir código en una sesión.
TIS	**Tiempo inactivo** durante una **sesión**.	Cantidad de tiempo empleado en investigación u ocio en una sesión.
NTC	**Número total** de **cambios**, es decir, acciones a nivel de desarrollador realizadas durante una sesión.	Cantidad de trabajo realmente realizado en una sesión.
TTC	**Tamaño total** de los **cambios**, es decir, número de cambios atómicos realizados durante una sesión.	Cantidad de trabajo realmente realizado en una sesión.
NRA	**Número** de **refactorizaciones** (registradas) y **acciones** relacionadas.	Corrección de errores.
NMA	**Número** de **métodos agregados** durante una sesión.	Nuevo comportamiento en el software.
NMM	**Número** de **métodos modificados** durante una sesión.	Nuevo comportamiento en el software.
NMD	**Número** de **métodos documentados**.	Código comprensible.
NCA	**Número** de **clases agregadas** durante una sesión.	Nuevo comportamiento en el software.
NCM	**Número** de **clases modificadas** durante una sesión.	Nuevo comportamiento en el software.
NCD	**Número** de **clases documentadas**.	Código comprensible.

2.3. Arquitectura Microkernel

Un plug-in o módulo es una pieza de software que se relaciona con una aplicación (llamada aplicación anfitrión o sistema central) para extender sus funcionalidades. El plug-in de por sí carece de la capacidad de ejecutar la funcionalidad que implementa, es por esto que depende del anfitrión para ejecutarse [3].

La arquitectura microkernel, usualmente conocida como arquitectura plug-in es un patrón de diseño que consta de dos componentes principales: el sistema central y los módulos. La clave principal de la arquitectura es proporcionar extensibilidad, flexibilidad y el aislamiento de las características de aplicación. El sistema central define cómo funciona el sistema y la lógica de negocio básica, mientras que los módulos son componentes independientes que contienen procesamiento especializado, características adicionales y código destinado a mejorar o ampliar el núcleo del sistema [1].

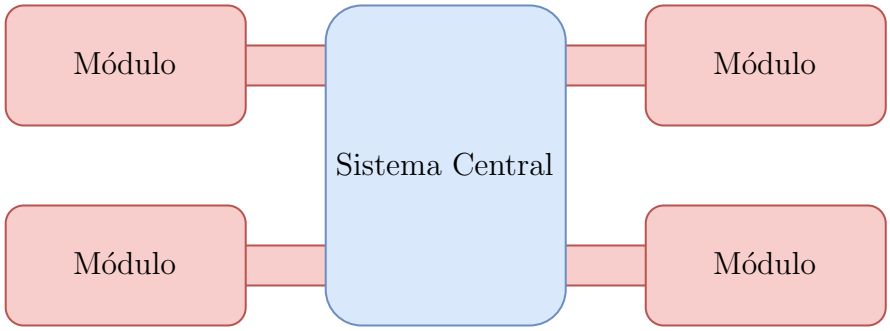


Figura 2.1: Patrón básico de la arquitectura Microkernel [1].

En general, los módulos deberían mantener al mínimo la comunicación con otros módulos, para evitar problemas de dependencia. Mientras que el sistema central se debería encargar de la lógica de negocio de la aplicación, reglas especiales y del procesamiento complejo que requiera el software [1].

Una ventaja de la arquitectura microkernel es que puede ser utilizada como parte de otro patrón de arquitectura. Por ejemplo, para implementar el módulo que se contempla para este proyecto se utilizará la arquitectura microkernel para empaquetar el patrón de diseño MVC (Modelo-Vista-Controlador), dejando el módulo disponible para ser usado por NetBeans.

2.4. Metodología de Desarrollo

En general las metodologías de desarrollo se encuentran diseñadas para ser utilizadas por equipos de personas enfocándose en el trabajo continuo y la entrega de resultados. Siguiendo las características que tiene el proyecto se aplicará una metodología ágil llamada PXP (Personal eXtreme Programming) la cual posee un subconjunto de prácticas presentes en XP (eXtreme Programming) siendo las más apropiadas para ser seguidas por un desarrollador autónomo. En PXP el proceso de desarrollo es iterativo y permite al desarrollador ser más flexible y receptivo a los cambios [18].

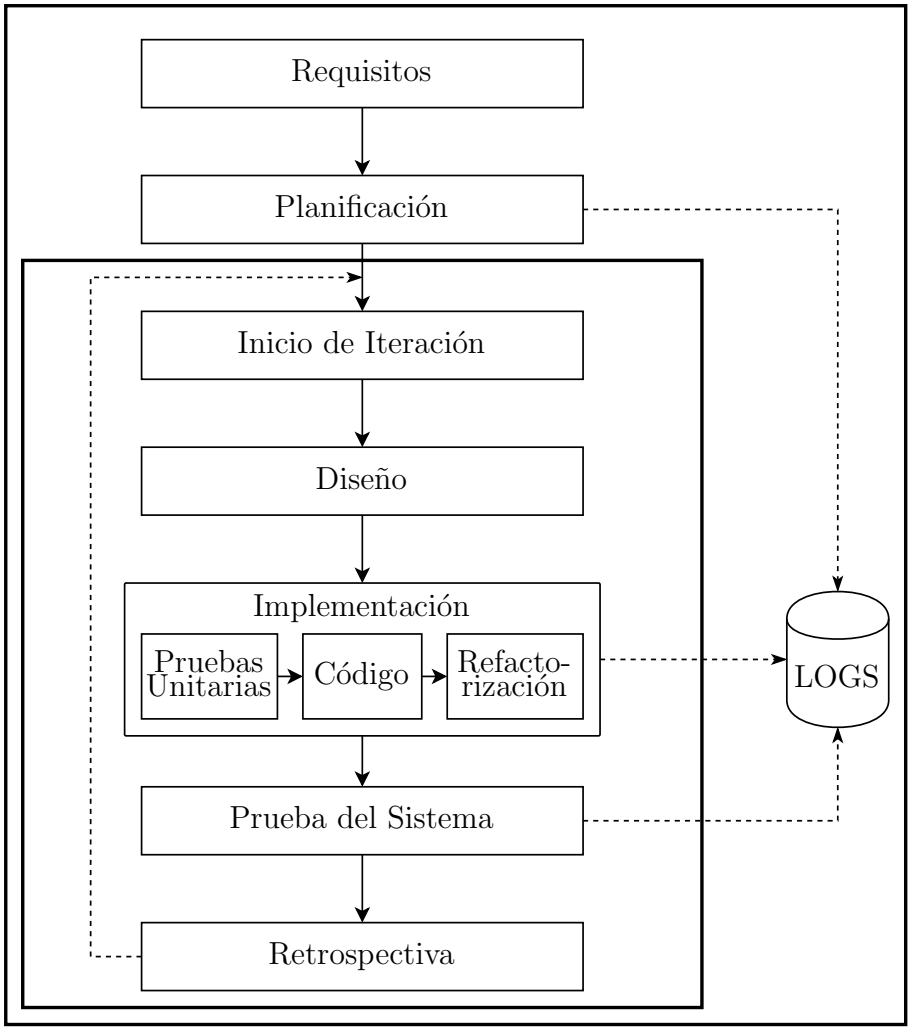


Figura 2.2: Fases de la metodología PXP.

En cuanto a las etapas de la metodología PXP en la figura 2.2 es posible visualizar el proceso de trabajo. A continuación, se entrega una descripción de lo que sucede en cada una de las fases [18]:

1. **Requisitos:** En esta etapa se crea un documento con los requisitos funcionales y no funcionales del proyecto definidas previamente con el cliente (en caso de existir uno). Estos requisitos deben tener una percepción general siguiendo los objetivos del proyecto.

2. **Planificación:** Aquí el desarrollador forma un conjunto de tareas basadas en el documento de requisitos donde cada una de ellas podría estar compuesta de tareas más pequeñas, a las que se le debe asignar una estimación de tiempo. La suma de las estimaciones de las tareas secundarias será la estimación de la tarea principal. Además, en esta fase se suelen definir las principales tecnologías y decisiones de diseño (lenguaje de programación, framework de desarrollo o la arquitectura de la aplicación).

3. **Iteraciones:**

    a) **Inicio de iteración:** Aquí se deben seleccionar las tareas a realizar considerando que la duración de cada iteración varía entre 1 a 3 semanas.

    b) **Diseño:** El desarrollador modela las clases que se implementarán durante la iteración. Quedando a cargo del desarrollador el método o herramienta de diseño a utilizar.

    c) **Implementación:** En esta fase se genera el código, el desarrollador implementa todos los objetos definidos en la fase anterior considerando también las pruebas unitarias. Para finalizar con esta etapa, el código debe compilarse sin errores y todas las pruebas unitarias deben pasar con éxito.

    d) **Prueba del sistema:** Aquí es donde se prueban todas las características desarrolladas hasta ahora, verificando si la solución implementada cumple con la definida por los requisitos del proyecto. Los defectos encontrados se registran y corrigen.

    e) **Retrospectiva:** Esta etapa marca el final de la iteración, aquí se realiza un análisis de los datos recopilados en las otras fases verificando que el tiempo de las tareas sea igual al real y así encontrar posibles retrasos.

Justificación para la elección de la metodología

Se decide trabajar con una metodología de desarrollo ágil, ya que están orientadas a generar software funcional y dejan de lado procesos excesivos de documentación o planificación de proyectos, al contrario que lo hacen las metodologías de desarrollo tradicional. De lo anterior, se decide trabajar con PXP ya que es una metodología apta para un proyecto individual considerando el contexto y problema planteado en el Capítulo 1. Solucionando el inconveniente que se tiene con Scrum o XP, metodologías que se encuentran diseñadas para equipos de más de 3 personas.

2.5. Metodología de evaluación

En cuanto a la evaluación del proyecto se usa de una metodología basada en experimentación, puesto que, al aplicarla en un proyecto relacionado con la ingeniería de software, ayuda a identificar y comprender distintas variables y conexiones que permiten validar o refutar con hechos las prácticas que se usaron para construir el proyecto. En consecuencia, el objetivo de la experimentación en ingeniería de software es poder disponer de soluciones que estén soportadas por un cuerpo de conocimientos validado por evidencia científica [19].

Usualmente, esta metodología se puede dividir en cuatro actividades principales que son [19]:

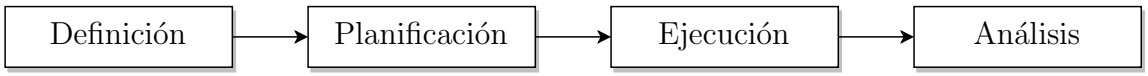


Figura 2.3: Actividades de la metodología basada en experimentación.

1. **Definición:** Consiste en delimitar de manera general los aspectos principales del experimento a llevar a cabo junto con precisar las hipótesis del trabajo.

2. **Diseño o planificación:** Aquí se especifica cómo se asignan los tratamientos a los sujetos, el tipo de sujetos a emplear y los instrumentos o materiales que se deben aplicar en el experimento.

3. **Ejecución:** En esta etapa es donde se lleva a cabo el experimento, antes de comenzar, los sujetos reciben una serie de instrucciones además de los materiales que utilizarán durante la sesión.

4. **Análisis:** En esta fase se recolecta una serie de métricas generadas por los sujetos. Las métricas son analizadas y comparadas con las hipótesis previamente definidas aplicando técnicas estadísticas para interpretar los datos obtenidos.

2.6. Herramientas para el desarrollo de software

En esta sección se mencionan y describen los lenguajes y herramientas que se utilizarán para la implementación de este proyecto. Contemplando que la herramienta necesita el módulo, la aplicación web y el servicio web para su funcionamiento.

2.6.1. Herramientas para la implementación del módulo

A continuación se detalla el lenguaje y herramienta a utilizar para implementar el módulo que se encargará de capturar y enviar al servidor web las distintas interacciones de usuario realizadas en NetBeans.

1. **Java**
   Java es un lenguaje de programación de alto nivel orientado a objetos. Es una tecnología que permite el desarrollo de programas que son capaces de ejecutarse en entornos distribuidos y heterogéneos. Java es un lenguaje de alto rendimiento, es multiplataforma, robusto, orientado a objetos, distribuido y concurrente. Características que otorgan una arquitectura flexible y robusta para el desarrollo de distintos programas [20].

2. **NetBeans**
   El entorno de desarrollo NetBeans IDE es una herramienta escrita en Java, gratuita y de gran utilidad para los programadores ya que permite escribir, compilar, depurar y ejecutar programas utilizando un gran número de lenguajes de programación, también incluye una tecnología de control y gestión de proyectos. Además, cuenta con un gran número de módulos que suelen mejorar la experiencia del usuario ya que extienden la funcionalidad del IDE [21].

2.6.2. Herramientas para la implementación de la aplicación web

Para la creación de la aplicación web que se utilizará para visualizar las métricas capturadas, se disponen a continuación las herramientas y lenguajes que se utilizarán.

1. **JavaScript**
   JavaScript es un lenguaje de programación dinámico que soporta la construcción de objetos basados en prototipos. Es utilizado principalmente en el desarrollo de páginas web, aunque también es posible desarrollar aplicaciones del lado del servidor. Las capacidades de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, entre otras [22].

2. **React**
   React es una biblioteca de JavaScript para construir interfaces de usuario. Se caracteriza por ser declarativo, permitiendo que el código sea predecible y fácil de depurar. Además, es basado en componentes, lo que significa que al crear estos componentes manejarán su propio estado, logrando crear interfaces de usuario complejas. React también se encarga de actualizar y renderizar de manera eficiente los componentes cuando los datos cambian [23].

2.6.3. Herramientas para la implementación del servicio web

A continuación se describen las herramientas y lenguajes utilizados para desarrollar el servicio web, el cual mantendrá una comunicación con la aplicación web y el módulo de NetBeans para enviar y recibir distintas respuestas.

1. **PHP**
   PHP es un lenguaje de código abierto adecuado para el desarrollo web del lado del servidor, permitiendo recibir las respuestas necesarias sin conocer el código subyacente. PHP se destaca por ofrecer una lista extensa de funcionalidades avanzadas que ayudan al programador en algunas tareas durante el desarrollo [24].

2. **Lumen**
   Lumen es un micro-framework basado en Laravel que facilita la escritura de servicios, obteniendo APIs sumamente rápidas en comparación a lo ofrecido por otras tecnologías. Además, conserva las características mas potentes usadas en Laravel como por ejemplo la validación, enrutamiento, middleware o la disponibilidad del ORM Eloquent [25] esencial y sencilla librería que es utilizada para comunicar la base de datos con los modelos definidos en el proyecto,

permitiendo la consulta de datos o la inserción de nuevos registros.

3. **MySQL**
   MySQL es una base de datos de código abierto que otorga rendimiento, fiabilidad y facilidad de uso. Convirtiéndose en la opción de base de datos más usada en aplicaciones basadas en la web [26].

Justificación de la elección de las herramientas

Para concluir, la decisión de utilizar NetBeans y Java para la construcción del módulo tiene estrecha relación con el contexto de este trabajo, ya que en cursos como Programación Avanzada o Proyecto de Programación de la carrera de Ingeniería Civil en Computación, se utilizan estas tecnologías para comprender el paradigma de programación orientada a objetos.

Por otra parte, para la construcción de la aplicación web, la principal razón para elegir las herramientas anteriormente descritas es la experiencia previa que se tiene al utilizar estas tecnologías. Además, según resultados obtenidos en la encuesta de programadores realizada por StackOverflow [27] en el año 2020, se da a conocer que tecnologías como React.js, Laravel y MySQL se encuentran muy bien posicionadas en comparación a su competencia, donde se destacan las categorías en que aparecen React y MySQL, las cuales se encuentran posicionadas en la segunda y primera posición, respectivamente.

3. Marco Metodológico

En este capítulo se explica detalladamente como se aplica la metodología Personal Extreme Programming (PXP) al proyecto actual, mostrando algunas modificaciones que se hacen a las etapas de PXP ajustándose al desarrollo del módulo y la aplicación web. Además, se habla de las herramientas utilizadas para manejar la planificación del proyecto.

3.1. Personal Extreme Programming (PXP)

La metodología PXP es útil ya que permite entregar aplicaciones funcionales en un tiempo relativamente rápido, además de que la fase de desarrollo se ajusta a un solo programador, permitiendo realizar ajustes a la metodología a conveniencia del desarrollo del proyecto, eliminando actividades innecesarias para un proyecto individual. En este sentido sucede algo similar con los roles que existen en este tipo de metodologías, ya que para PXP y este proyecto es posible definir sólo dos roles, detallados a continuación:

1. El **cliente** describe todas las necesidades deseables para la aplicación a desarrollar, determinando además su prioridad. Durante el desarrollo suelen ocurrir cambios y ajustes de requisitos, por lo que, es importante que mantenga una comunicación estrecha con el desarrollador.

2. El **desarrollador** participa en todas las etapas del desarrollo: se comunica con el cliente, define la arquitectura de la aplicación y constantemente debe programar, probar y refactorizar código. Por esto, es esencial contar con una planificación y llevar nota de las decisiones, cambios o problemas que surgen en el desarrollo.

3.2. Requisitos

Esta etapa esta destinada a obtener los requisitos funcionales del proyecto a desarrollar. Para esto, durante las dos primeras semanas del desarrollo, se concretaron reuniones con el cliente enfocadas a recopilar las funcionalidades de la herramienta. Para este proyecto, las necesidades se capturan utilizando historias de usuario ya que estas son [28]:

1. **Negociables:** La historia en si misma no se considera un contrato, siendo necesaria la constante discusión con el cliente para esclarecer su alcance.

2. **Estimables:** Para establecer la historia de usuario es necesario tener la estimación del tiempo que llevará completarla. permitiendo estimar el tiempo total del proyecto.

3. **Verificables:** Las historias de usuario cubren requerimientos funcionales, por ende son verificables. Lo ideal, es que puedan ser verificadas en cada entrega del proyecto.

El formato de escritura que se sigue para las historias de usuario es 'Como [*rol*] necesito [*algún objetivo*] para poder [*alguna razón*]'. Las historias se anotan en un documento llamado Pila de Producto donde se define para cada registro un ID, un Enunciado de la historia, un Alias para la historia, un Estado (Pendiente, En proceso, Pruebas o Completado), el Esfuerzo (Tiempo estimado), el Sprint en que se realiza y la Prioridad asignada.

3.3. Planificación

En esta etapa se define de manera general las Iteraciones o Sprints a cumplir durante el desarrollo del proyecto, se determina que cada Iteración debe durar 3 semanas. También se define como se realiza la estimación o esfuerzo de cada tarea concluyendo en que se debe hacer una suposición del tiempo aproximado que costará desarrollarla. En la Tabla 3.1 se especifica la planificación de cada Iteración:

Tabla 3.1: Planificación de Iteraciones.

Fecha de Inicio	Fecha de Termino	Iteración	Objetivo general
09 de Septiembre de 2019	30 de Septiembre de 2019	1	Definir arquitectura lógica y física del módulo y la aplicación web. Comenzar el desarrollo del módulo.
30 de Septiembre de 2019	21 de Octubre de 2019	2	Definir la forma en que el módulo valida al estudiante. Capturar eventos de inserción y eliminación.
21 de Octubre de 2019	11 de Noviembre de 2019	3	Capturar interacciones relevantes de una sesión. Enviar información recopilada a un servicio web.
11 de Noviembre de 2019	01 de Diciembre de 2019	4	Definir diagrama de Base de Datos para el módulo. Implementar servicio web y base de datos para almacenar las sesiones.
01 de Marzo de 2020	21 de Marzo de 2020	5	Definir vistas preliminares para la aplicación web.
21 de Marzo de 2020	10 de Abril de 2020	6	Definir diagrama de Base de Datos para la aplicación web. Configurar el ambiente de desarrollo para la aplicación web.
10 de Abril de 2020	30 de Abril de 2020	7	Autenticar y definir permisos para los usuarios de la aplicación web. Definir la gestión de organizaciones, usuarios y cursos en la aplicación web.
30 de Abril de 2020	18 de Mayo de 2020	8	Definir la gestión de instancias de cursos, proyectos y estudiantes en la aplicación web.
18 de Mayo de 2020	06 de Junio de 2020	9	Definir la gestión de grupos para los proyectos de la aplicación web. Carga masiva de estudiantes. Visualización de Métricas de desarrollo.
06 de Junio de 2020	24 de Junio de 2020	10	Correcciones para la aplicación web y el módulo. Despliegue de la aplicación web y el módulo.

Para llevar registro de lo planificado se hace uso de un tablero Kanban ya que permite hacer seguimiento visual del flujo de trabajo. Para este propósito, se utiliza Trello ya que permite crear de forma rápida un tablero Kanban digital. En este tablero es posible crear listas útiles para organizar tarjetas que contienen tareas a realizar, las tarjetas se van moviendo por las listas a medida que se ponen en cola, se trabaja en ellas y se completan [29]. La estructura de la tarjeta Trello mostrada

en la Figura 3.1 que se adopta para este proyecto es principalmente el nombre de la tarea y una etiqueta que la categoriza. De manera opcional, se puede incorporar una descripción de la tarea, una fecha de vencimiento, un checklist con funcionalidades aun más específicas o una imagen.

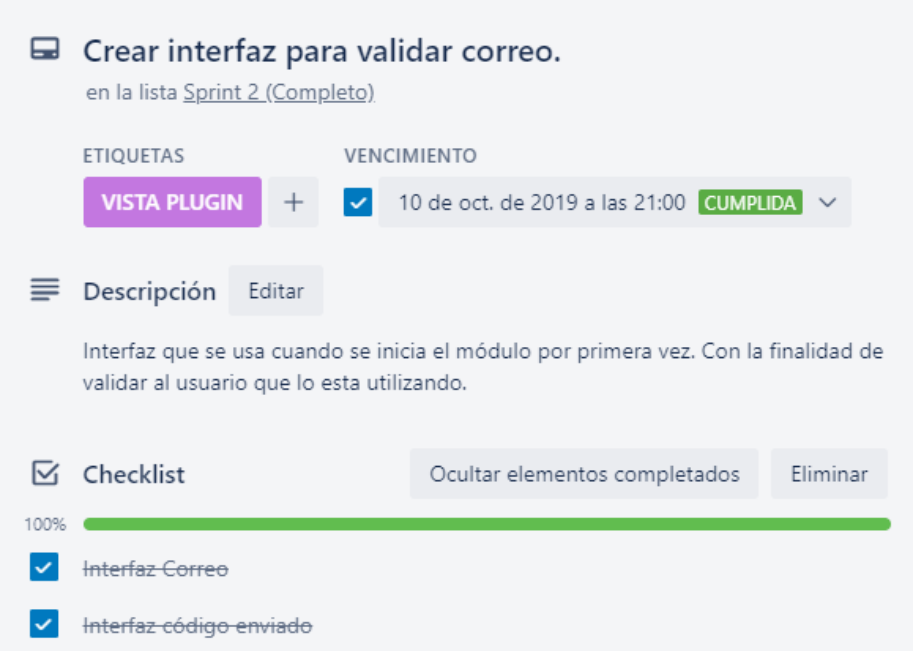


Figura 3.1: Ejemplo de una tarjeta Trello.

Las tarjetas se van moviendo por las diferentes listas a medida que el proyecto avanza. Estas listas son columnas que representan una actividad específica que, en conjunto, conforman el flujo de trabajo [29]. En la Figura 3.2 se visualiza de manera general el flujo básico que se tiene para este proyecto. A continuación, se describe la utilidad de cada columna:

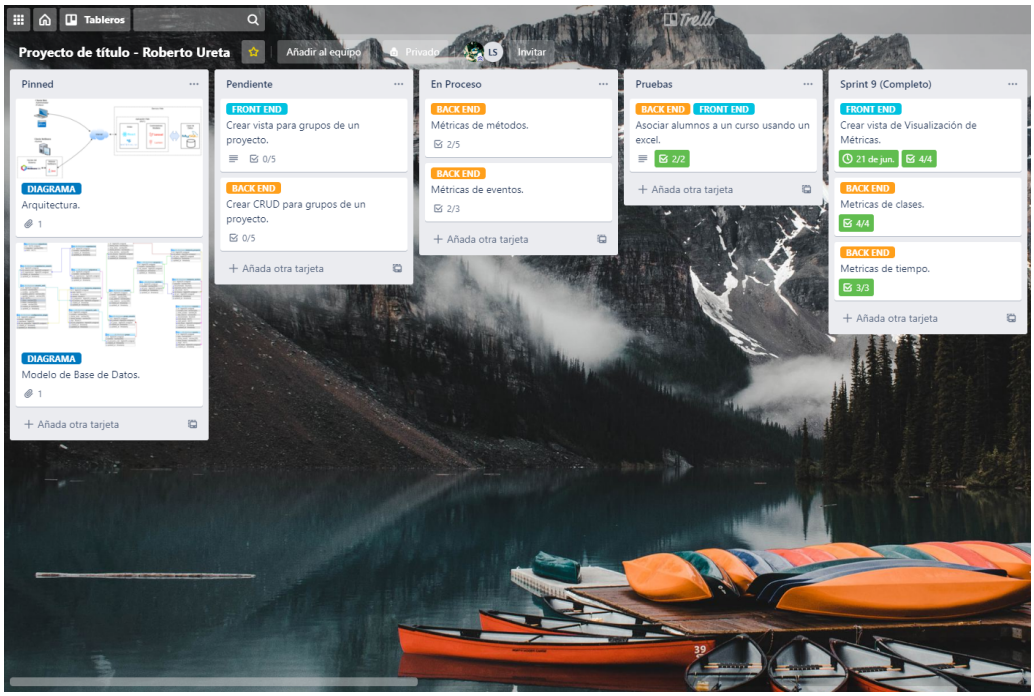


Figura 3.2: Flujo de trabajo utilizado en el proyecto.

1. **Fijado:** En esta columna se mantienen las tarjetas con información relevante para el proyecto, son tarjetas que periódicamente son consultadas para resolver distintas dudas que surgen durante el desarrollo. Para ilustrar, en esta lista se almacenan tarjetas que contienen la Arquitectura de la herramienta o el Diagrama relacional de la Base de Datos.

2. **Pendiente:** Esta columna se completa al principio de cada iteración, ya que contiene las distintas tareas que se deben implementar durante el avance de la misma. Si hay tareas que se intentan desarrollar y por algún motivo no se completan, deben volver a esta lista para continuar con su implementación en la siguiente iteración.

3. **En Proceso:** En esta columna se mantienen todas las tareas que están en proceso de desarrollo. Lo ideal es trabajar en una tarea, completarla y avanzar a la siguiente pero no existen restricciones para esto por lo que se puede trabajar en varias tareas de manera simultánea.

4. **Pruebas:** Esta columna alberga tareas que son completadas pero deben ser revisadas en conjunto en la etapa de prueba del sistema. Si se encuentran fallos

que no se alcanzan a solucionar en la iteración actual, las tareas deben volver a la lista de Pendiente para ajustarlas en la siguiente iteración.

5. **Sprint Completo:** Para cada iteración existe una columna con este nombre, con la finalidad de acceder e identificar de forma sencilla las tareas que fueron completadas en cada iteración. Para que una tarea se considere completada, debe pasar por los procesos de implementación y prueba del sistema descritos en la siguiente sección.

3.4. Iteraciones

Como su nombre lo indica, esta etapa es iterativa y se contempla un máximo de tres semanas por ciclo. Se compone de las fases más importantes de la metodología PXP para conseguir un producto funcional al terminó de cada iteración, a continuación se describe como se implementa cada una de estas fases.

3.4.1. Inicio de iteración

En esta etapa, según la prioridad se elijen las historias de usuarios a desarrollar, estas se obtienen del documento que contiene la Pila de Producto. En este mismo documento se define una lista de pendientes, con las tareas creadas en base a cada historia de usuario. Esto se realiza en cada iteración. En esta lista se indica el ID de la historia de usuario a trabajar, el Alias de la historia de usuario, el enunciado de la tarea, el estado (Pendiente, En proceso, Pruebas o Completo), el Costo Estimado y el tiempo trabajado dividido en las semanas de duración de la iteración.

3.4.2. Diseño

Esta etapa se destina a definir los patrones de diseño que se siguen a la hora de implementar código y generar diagramas útiles para el desarrollo del proyecto, estos son: los diagramas para la Arquitectura física y lógica de la herramienta, el diagrama relacional de base de datos y los diagramas de secuencias. La utilidad principal de los diagramas es orientar al cliente y al desarrollador a lo largo del proyecto. En general, esta etapa requiere más tiempo en las primeras iteraciones del proyecto porque es esencial diseñar una arquitectura que cumpla y se adapte al proyecto, en las siguientes iteraciones se utiliza para realizar modificaciones menores a los distintos diagramas.

3.4.3. Implementación

En esta etapa, se hace una modificación a lo que dicta la metodología PXP. Alterando la fase de pruebas unitarias, debido a que es necesario aprender tres nuevas tecnologías que cumplan con este propósito. Por ejemplo, JUnit para NetBeans, React Testing Library para Reactjs y PHPUnit en el caso de Laravel. En vista del corto tiempo disponible para desarrollar el proyecto, se decide hacer algunas modificaciones a esta fase en el proceso de implementación. De esto, las fases comprendidas son las siguientes:

1. **Pruebas Unitarias:** En esta etapa se simplifica el uso de herramientas para realizar pruebas unitarias debido a la gran inversión de tiempo que conlleva aprender a usarlas, la complejidad que existe al probar algunos componentes y los ajustes que se deben hacer a estas pruebas cada vez que se modifica alguna función.

   Para solventar esto, en cada función y clase de la aplicación web se depura el código de manera exhaustiva para asegurar su correcto funcionamiento. Además, se utiliza Postman [30] para probar el funcionamiento de la API y la extensión React Developers Tools para revisar las jerarquías de los componentes de React.

2. **Código:** Consiste en la codificación de las tareas seleccionadas para la iteración. Esta es la etapa que consume el mayor tiempo de toda la iteración, debido a que algunas tareas requieren más tiempo que el asignado o porque aparecen problemas en la implementación.

3. **Refactorización:** Una vez que la funcionalidad está totalmente disponible se procede a refactorizar el código, para conseguir que sea entendible y reutilizable. Este proceso consigue que el desarrollo sea robusto y menos susceptible a fallos, asegurando su mantenimiento en el futuro.

3.4.4. Prueba del sistema

Esta etapa consiste en validar el funcionamiento del sistema completo enfocándose en las características agregadas recientemente. Para continuar con la siguiente iteración, estas características deben cumplir lo solicitado por la historia de usuario. Para validar lo anterior, se aplican pruebas de caja negra[1]. Si se encuentra algún fallo, se debe solucionar en esta etapa con la finalidad de entregar un prototipo funcional. Las características incompletas no se agregan a este prototipo, en consecuencia se mueven a la siguiente iteración para ser completadas al principio de esta.

3.4.5. Retrospectiva

Esta etapa marca el final de cada iteración, participan el desarrollador y el cliente del proyecto para revisar el prototipo funcional generado y la Pila de Producto con los objetivos planteados para la iteración. Se utiliza para determinar si los tiempos que lleva realizar cada tarea son similares a los tiempos inicialmente estimados, con la finalidad de ajustar las estimaciones a medida que avanza el proyecto. Además, esta fase es de utilidad para detectar contratiempos o para aplicar mejoras de distinta índole al desarrollo del proyecto.

[1]Es un método de pruebas en que un componente recibe una entrada para que retorne una salida esperada, sin la necesidad de conocer su implementación interna[31].

4. BlueLogger

El objetivo de este capítulo es mostrar a detalle el desarrollo contemplado para la propuesta. Comienza con la concepción del proyecto que da a conocer al propulsor de la idea, la problemática y la forma en que se estructura la solución. Luego, se da paso a la implementación de la solución donde se da a conocer el listado de historias de usuario de este proyecto, para luego enseñar el flujo de trabajo que se sigue en las distintas iteraciones, adoptando la metodología de desarrollo PXP.

4.1. Concepción del proyecto

La idea del proyecto surge en conversaciones con el profesor Luis Silvestre (docente de la Universidad de Talca), en las que comenta sus intenciones de tener un *plugin* que sea capaz de obtener distintas interacciones de usuario al momento de programar en un IDE como NetBeans, con la finalidad de obtener métricas de desarrollo. Él tiene conocimiento de *plugins* similares disponibles para el IDE Eclipse. Lamentablemente y en base a la experiencia obtenida de los distintos cursos de la carrera Ingeniería Civil en Computación, este entorno de desarrollo tiene bajo uso en estudiantes. Esto se puede justificar porque, en los cursos de programación de segundo año de la carrera se suele utilizar NetBeans, ya que este IDE cuenta con una menor curva de aprendizaje.

Desde el jueves 5 de Septiembre de 2019 se comienza a concretar reuniones semanales con el profesor. En un principio las reuniones son destinadas a definir las características del nuevo proyecto, principalmente se discuten las limitantes que tendrá el *plugin*. En dichas reuniones se decide además, que se debe implementar un prototipo

para una aplicación web encargada de mostrar las distintas métricas de desarrollo basándose en las interacciones capturadas por el *plugin*. Con esto en cuenta, se decide dividir la proyecto en dos partes:

1. **Módulo para NetBeans**
   El módulo es el encargado de recopilar y enviar a un servidor las distintas interacciones realizadas principalmente por estudiantes en los proyectos donde se utiliza el IDE NetBeans.

2. **Aplicación Web**
   La aplicación web es la encargada de gestionar las organizaciones, cursos, proyectos y alumnos. Destinando su uso principalmente a profesores para que visualicen las métricas calculadas en base a lo recopilado por el módulo, permitiéndoles tomar ciertas decisiones en base a esta información.

4.2. Historias de usuario

En las dos primeras reuniones con el profesor, se definen las historias de usuario que contempla la herramienta a desarrollar. En la Tabla 4.1 se pueden apreciar las historias destinadas al módulo para NetBeans, mientras que en la Tabla 4.2 se aprecian las historias destinadas a la aplicación web.

Tabla 4.1: Historias de usuario para el módulo de NetBeans.

ID	Enunciado de la historia
E001	Como cliente, necesito identificar a un programador al iniciar sesión para registrar sus sesiones.
E002	Como cliente, necesito recopilar interacciones básicas de interacción de un programador para identificar métricas de desarrollo.
E003	Como cliente, necesito recopilar interacciones avanzadas de un programador para identificar métricas de desarrollo.
E004	Como cliente, necesito recopilar eventos producidos por un programador para identificar cuando se copia o elimina código.
E005	Como cliente, necesito guardar las sesiones de programación de un usuario de NetBeans para consultarlas en otro momento.
E006	Como cliente, necesito tener información sobre los paquetes creados en un proyecto para identificar posibles cambios de los archivos.
E007	Como cliente, necesito tener información sobre las clases de un proyecto para identificar modificaciones en los archivos.
E008	Como cliente, necesito almacenar el registro de los eventos e interacciones en un archivo para guardarlos localmente.
E009	Como cliente, necesito enviar los archivos de registro a un sistema externo mediante internet para evitar que se pierda la información.
E010	Como cliente, necesito mostrar el estado de actividad del plugin para que el usuario que lo utiliza sepa si el plugin esta funcionando.

Tabla 4.2: Historias de usuario para la aplicación web.

ID	**Enunciado de la historia**
E011	Como cliente, necesito que el sistema permita iniciar sesión para identificar a los distintos usuarios de la aplicación web.
E012	Como cliente, necesito que el sistema permita la creación de cursos para que los profesores lleven un control sobre estos.
E013	Como cliente, necesito que el sistema permita crear proyectos asociados a un curso para que el profesor realice el seguimiento de los alumnos.
E014	Como cliente, necesito que el sistema permita asociar alumnos a un determinado proyecto para definir grupos de trabajo.
E015	Como cliente, necesito que el sistema permita visualizar las métricas de desarrollo de un determinado programador para que los profesores puedan tomar decisiones.
E016	Como cliente, necesito que el sistema guarde todos los archivos que han sido enviados desde el plugin para consultarlos desde la aplicación web.
E017	Como cliente, necesito que el sistema permita la gestión de Organizaciones para administrar cursos y profesores de otras universidades.
E018	Como cliente, necesito que el sistema permita la gestión de usuarios en la aplicación web para identificar roles de Administrador y de Profesor.

4.3. Desarrollo de la solución

En primer lugar, se describe el desarrollo comprendido para la herramienta propuesta BlueLogger, siguiendo las etapas dispuestas en el apartado de iteraciones que dicta la metodología de desarrollo PXP. A continuación, se dan a conocer cada una de las iteraciones donde se explica a detalle la iteración uno con la finalidad de exponer el cumplimiento de la metodología. Para las siguientes iteraciones se expone un resumen de lo desarrollado dando a conocer los objetivos a lograr, las funcionalidades implementadas, los diagramas o vistas importantes para la comprensión del proyecto y por último el estado final del prototipo.

4.3.1. Iteración 1

El objetivo de esta iteración es definir el diagrama para la arquitectura física y lógica del módulo, y la aplicación web. Además, se comienza a desarrollar el módulo en el ambiente NetBeans. En cada etapa de la metodología PXP se realiza lo siguiente:

1. **Inicio de Iteración:** Las tareas seleccionadas corresponden a obtener información del proyecto y las clases asociadas al usuario de NetBeans que se encuentra desarrollando. La historia involucrada es la **E002**.

2. **Diseño:** Se definen los diagramas de arquitectura física y lógica para el módulo y la aplicación web. La Figura 4.1 corresponde al diagrama de la arquitectura lógica del módulo que sigue lo expuesto por la arquitectura micro-kernel, permitiendo empaquetar el patrón de diseño Modelo-Vista-Controlador. Los demás diagramas se encuentran disponibles en el **Anexo A.1**.

3. **Código:** Se comienza con la creación del módulo compatible con el IDE NetBeans. Principalmente se trabaja en la implementación de un *Listener* [1] encargado de obtener el nombre y ruta del Proyecto o las clases que se tienen abiertas en el editor de NetBeans en el momento en que un usuario trabaja en ellas.

[1] Los *Listeners* son métodos que se invocan cuando el usuario hace algo (por ejemplo, hacer click en un botón) que el programador requiera que sea notificado [32].

4. **Pruebas Unitarias:** Se depuran los métodos encargados de crear y obtener un objeto para el Proyecto y los archivos abiertos en el editor de código de NetBeans. Para esto se prueba con archivos de distintos proyectos realizados en Java imprimiendo la ruta y nombre de estos.

5. **Refactorización:** Se separan los métodos *Listeners* que obtienen el proyecto y los archivos de la clase que inicia el módulo y son llevados a otra clase que los implementa, buscando separar la lógica del módulo.

6. **Prueba del Sistema:** Para probar el módulo se inicia otra instancia del IDE NetBeans que incluye el módulo desarrollado revisando principalmente que no se presenten fallos a la hora de utilizar el editor de código.

7. **Retrospectiva:** En esta primera iteración se disponen de algunos diagramas que dan a conocer como funciona la herramienta. Además, se logra generar un módulo funcional que es capaz de obtener información de proyectos y clases abiertas en el editor de código del IDE.

   La tarea relacionada con capturar el tiempo inicial de una sesión de programación se bloquea para la siguiente iteración porque es algo que requiere de más tiempo para ser implementado.

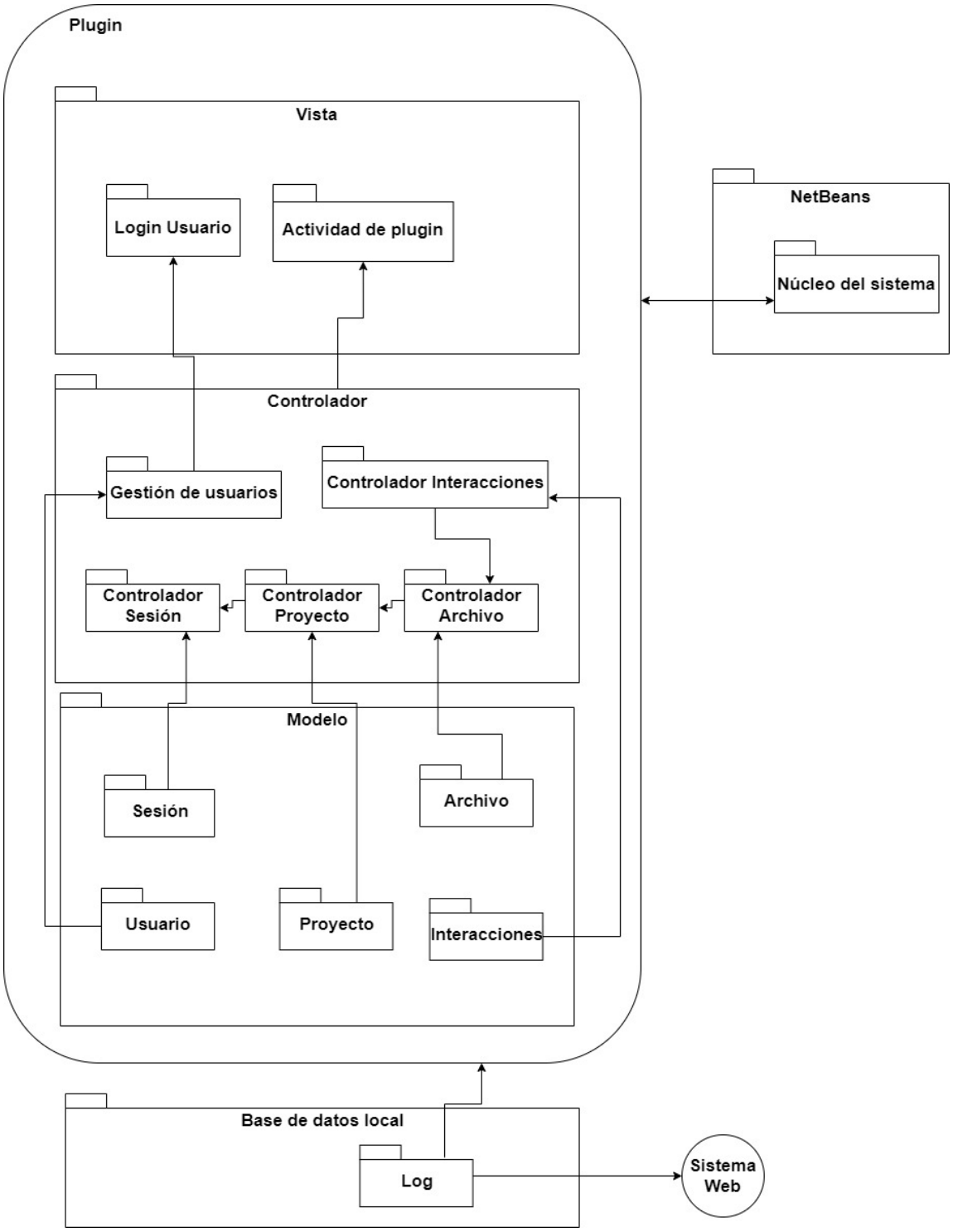


Figura 4.1: Arquitectura Lógica del Módulo para NetBeans.

4.3.2. Iteración 2

El objetivo de esta iteración es validar al usuario que utiliza el módulo en NetBeans, capturar eventos de inserción o eliminación y definir todas las interfaces de usuario que tiene el módulo. Las historias de usuario involucradas son **E001**, **E004** y **E010**.

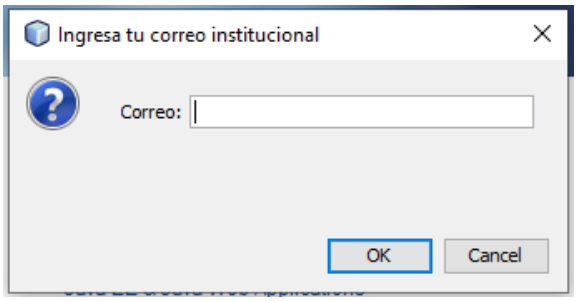
Para validar al usuario, se dispone de una ventana donde se solicita un correo para enviar un código de validación. Luego, el usuario lo debe copiar y pegar en la siguiente ventana visualizada en la Figura 4.3b. Cuando se hace esto de forma correcta se crea un archivo de configuración para que el usuario no tenga que repetir esto cada vez que se inicia NetBeans.

Por último, se agrega la funcionalidad para detectar eventos, estos suceden cuando se inserta o elimina una selección de código que supere los 100 caracteres considerando que este número debiese ser el límite de caracteres para una línea de código, según la Guía de estilo de Google para Java [2]. La información que se captura esta disponible en la Figura 4.2.

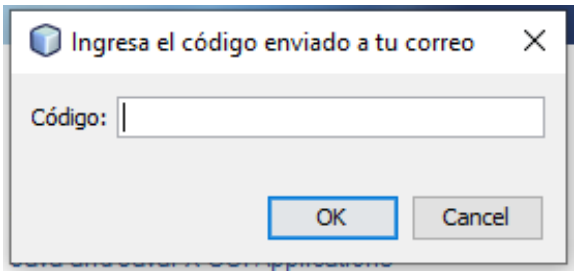
```
1 "eventos": [
2     {
3       "fecha": "10/06/2020",
4       "hora": "18:01:22",
5       "largo": 184,
6       "tipo": "INSERT"
7     }
8 ]
```

Figura 4.2: JSON con la información capturada referente a los eventos de una clase.

[2]Número de caracteres máximo por linea de código: `https://google.github.io/styleguide/javaguide.html#s4.4-column-limit`



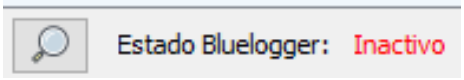
(a) Correo.



(b) Código.

Figura 4.3: Validación de usuario en NetBeans.

(a) Bluelogger iniciando.



(b) Bluelogger inactivo.

(c) Bluelogger funcionando.

Figura 4.4: Estados de funcionamiento del módulo en NetBeans.

Estado del prototipo: Como lo muestran las Figuras 4.3 y 4.4 se cuenta con interfaces de usuario que posibilitan la visualización del módulo en NetBeans. También, es posible identificar al usuario que utiliza el módulo. Por último, se detecta cuando se inserta o elimina código con más de 100 caracteres.

4.3.3. Iteración 3

El objetivo de esta iteración es obtener interacciones avanzadas de los usuarios que utilizan el módulo, detectar el tiempo activo e inactivo de un usuario al programar en

NetBeans y enviar la información recopilada a un servidor. Las historias de usuario involucradas son **E003, E005, E006, E007** y **E008**.

En esta iteración se obtiene información relevante de las clases visualizadas por el usuario de NetBeans. Por ejemplo, para cada una de las clases se obtiene su nombre, la cantidad de líneas que tiene el archivo, si la clase esta documentada, el paquete al que pertenece y los segundos activos en que el usuario trabajo en ella. También, se obtiene información de los métodos asociados a cada clase, por cada método se guarda su nombre, su nivel de acceso, el tipo de retorno, si esta documentado, el número de líneas y el número de parámetros que tiene. Lo anterior se visualiza claramente en la Figura 4.5.

```
{
  "archivos": [
    {
      "nombreArchivo": "NombreClase",
      "paquete": "nombrePaquete",
      "cantLineasArc": 35,
      "documentado": false,
      "segundosActivo": 54,
      "metodos": [
        {
          "nombre": "nombreMetodo",
          "nivelAcceso": "PRIVATE",
          "tipoRetorno": "String",
          "documentado": true,
          "cantLineas": 10,
          "cantParametros": 2
        }
      ]
    }
  ]
}
```

Figura 4.5: JSON con la información capturada referente a los métodos de una clase.

Además, se captura el tiempo total de la sesión de un usuario guardando dos marcas de tiempo, una cuando se abre NetBeans y otra cuando se cierra NetBeans. Por último, se implementa la forma en que un objeto JSON que contiene toda la información recopilada debe ser enviado al servidor, con esto se determina que, si el servidor no está disponible el objeto JSON queda guardado en el equipo del usuario. La estructura completa de este JSON esta disponible en el **Anexo A.2**.

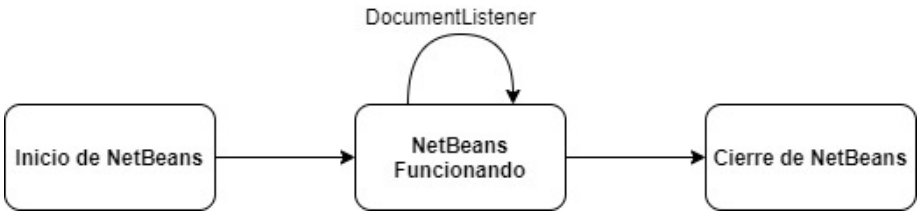


Figura 4.6: Ciclo de vida de NetBeans.

Para conseguir lo descrito anteriormente, se utiliza el ciclo de vida de NetBeans visualizado en la Figura 4.6 para realizar distintas tareas en tres de sus etapas. Algunas de estas tareas son:

1. **Inicio de NetBeans:**
   - Verificar existencia de archivo de configuración.
   - Actualizar estado de BlueLogger.
   - Obtener una marca de tiempo que indica el inicio de NetBeans.
   - Agregar *DocumentListener* para la siguiente etapa.
2. **NetBeans Funcionando:** El *DocumentListener* se encarga de
   - Guardar los proyectos y archivos visualizados.
   - Guardar los eventos de Inserción y Eliminación que superen los 100 caracteres.
   - Calcular el tiempo activo por archivo.
3. **Cierre de NetBeans:**
   - Obtener marca de tiempo que indica el cierre de NetBeans.

- Obtener información de los archivos visualizados.
- Enviar sesión al servidor.

Estado del prototipo: En este punto ya se encuentran concluidas todas las historias de usuario relacionadas con el módulo. Este ya es completamente funcional, ya que es posible obtener información de los proyectos, clases y métodos en que un usuario trabaja.

Además, al utilizar el ciclo de vida de NetBeans, es posible obtener el tiempo total de la sesión de desarrollo, incluyendo también el tiempo invertido en cada clase visualizada. También, la información recopilada puede ser enviada a un servidor y en caso de no existir conexión se puede almacenar de manera local en el computador del usuario.

4.3.4. Iteración 4

El objetivo de esta iteración es crear la lógica del servicio backend y definir el modelo de base de datos que contendrá la información recopilada por el módulo. Las historias de usuario involucradas son **E008** y **E009**.

Para comenzar, se define el modelo de datos disponible en la Figura 4.7. Su diseño esta orientado para guardar la información recopilada por el módulo, donde cada usuario puede trabajar en distintos proyectos en una misma sesión, además de los recurrentes cambios en los mismos archivos a lo largo de varias sesiones.

Luego, se implementa un servicio web basado en *API REST* usando la tecnología Node.js. En este servicio, se define un *endpoint* encargado de recibir la información recopilada por el módulo, registrándola en la base de datos.

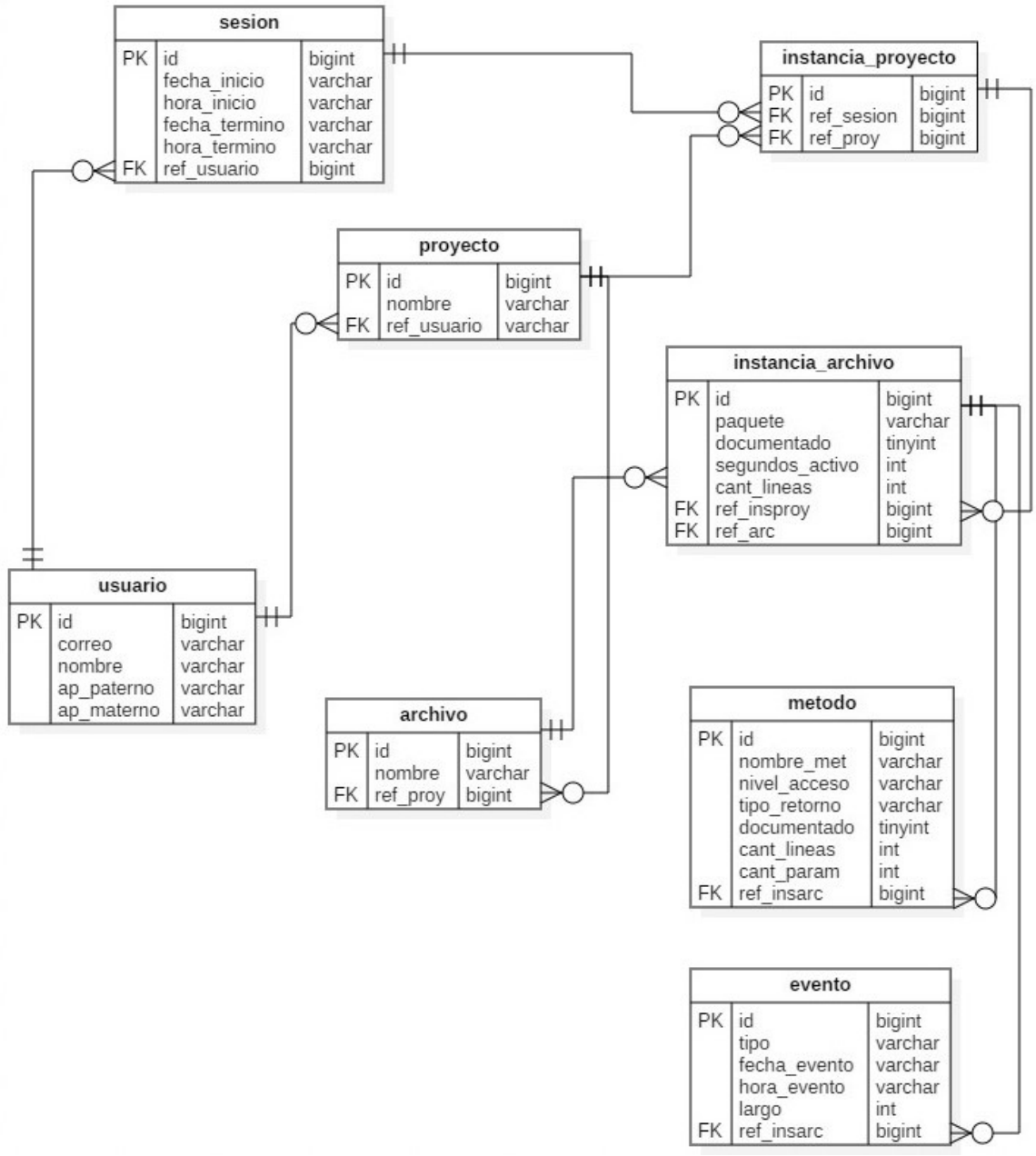


Figura 4.7: Modelo de datos para las sesiones enviadas por el módulo.

Estado del prototipo: El módulo ya es capaz de enviar la información recopilada al servicio web definido, este último ya está configurado y cuenta con lo necesario para almacenar las distintas sesiones producidas por los usuarios del módulo.

4.3.5. Iteración 5

El objetivo de esta iteración es migrar el servicio web de Node.js al micro-framework Lumen de Laravel, además se trabaja en la creación de los *mockups* preliminares para la aplicación web. La historia de usuario involucrada es la **E016**.

El principal motivo para migrar el servicio web fue que las librerías ocupadas en Node.js generaban muchos problemas con la conexión a la base de datos. Es por esto, que se cambia la tecnología usada por una mas robusta y confiable como lo es Laravel. Además, hasta este punto sólo se contaba con el *endpoint* que recibe las sesiones del módulo. Por lo que no fue una tarea complicada.

Luego, se comienza a trabajar en las vistas preliminares de la aplicación web. Se crean las vistas que visualizara el profesor, estas son: las Organizaciones a las que pertenece, las Asignaturas que imparte, los Alumnos/Proyectos de una Asignatura, el Proyecto y la visualización de métricas. En las Figuras 4.8 y 4.9 se muestran algunos ejemplos de los *mockups* realizados. El resto de interfaces de usuario están disponibles en `https://bluelogger.bss.design`, la PassKey de acceso es 1234.

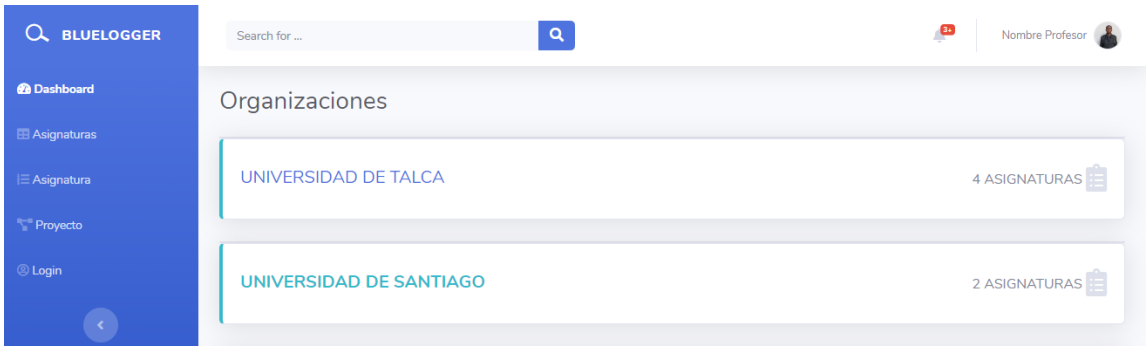


Figura 4.8: Vista Preliminar de Organizaciones.

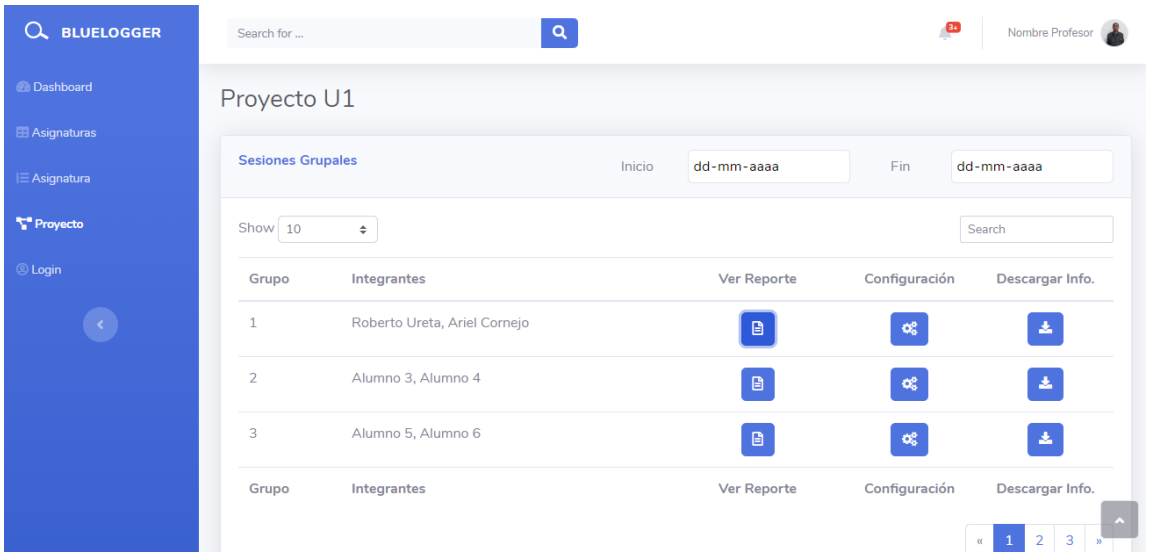


Figura 4.9: Vista Preliminar de un Proyecto.

Estado del prototipo: Se cambia de manera definitiva el servicio web definido para recibir las sesiones del módulo. De esta forma, en el diagrama de secuencia de la Figura 4.10 se aprecian todos los participantes que interactúan para generar, obtener y almacenar las distintas sesiones de programación.

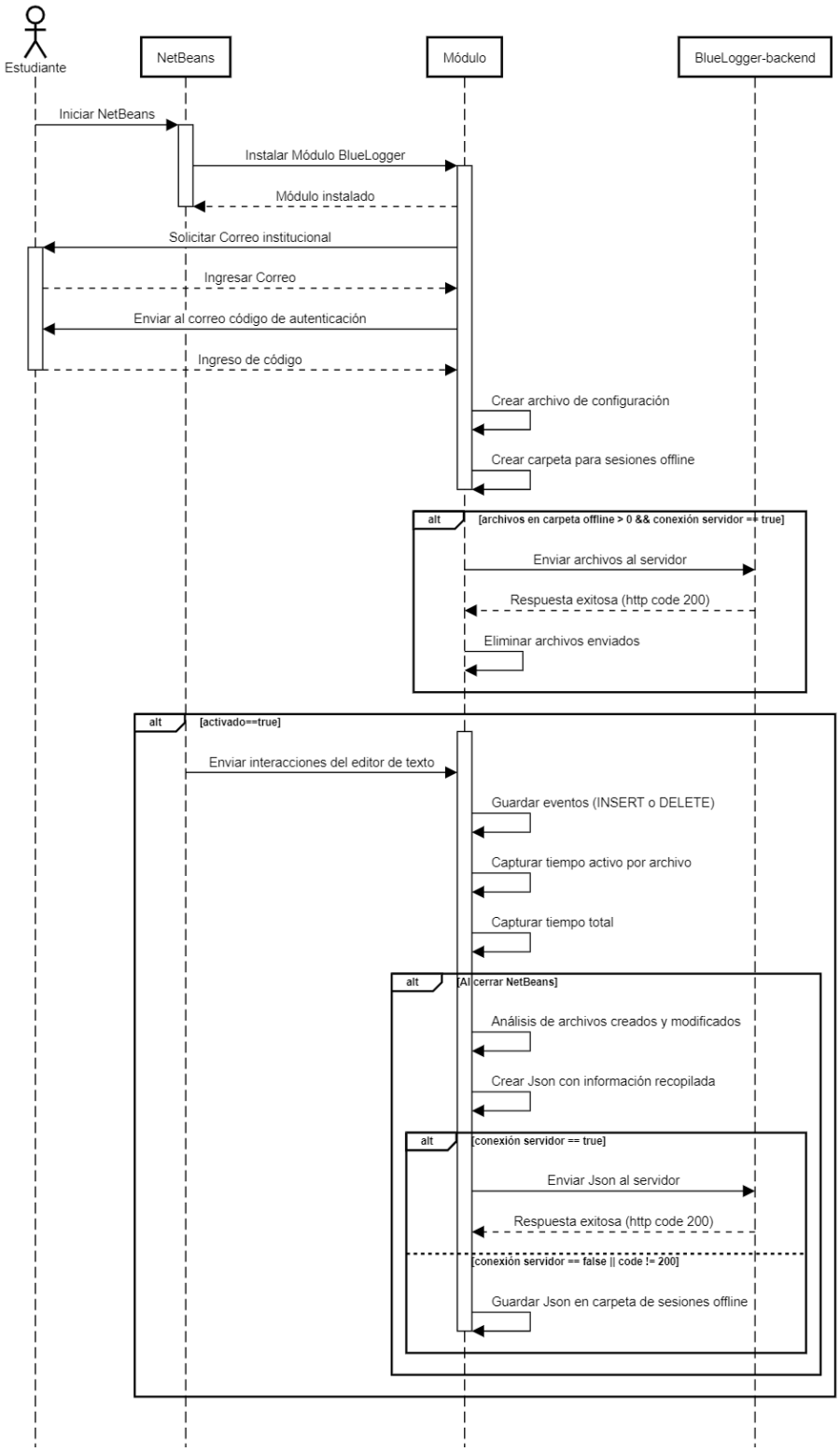


Figura 4.10: Diagrama de Secuencia del módulo.

4.3.6. Iteración 6

El objetivo de esta iteración es comenzar con la implementación de la aplicación web en React, definiendo el modelo de datos y los principales componentes de la web. Se consideran todas las historias de usuario relacionadas con la aplicación web con la finalidad de construir el modelo de datos.

Primero, se define el modelo de datos a utilizar en la aplicación web visualizado en la Figura 4.12, de esto se destaca que la entidad usuario (estudiante que utiliza el módulo) corresponde a la presentada anteriormente en la Figura 4.7, conectando ambos modelos.

Luego, se comienza a utilizar React para el desarrollo de la aplicación web, teniendo esto en cuenta, la Arquitectura Física de la herramienta queda expuesta en la Figura 4.11. Para la comunicación entre la aplicación web y la *API REST*, en React se define un *dataProvider* (solicitudes GET, POST y PUT) y un *authProvider* (autenticación del usuario web) para convertir las diferentes llamadas de los componentes en solicitudes HTTP adaptando las respuestas a un formato estándar conocido por todos los componentes de la aplicación web.

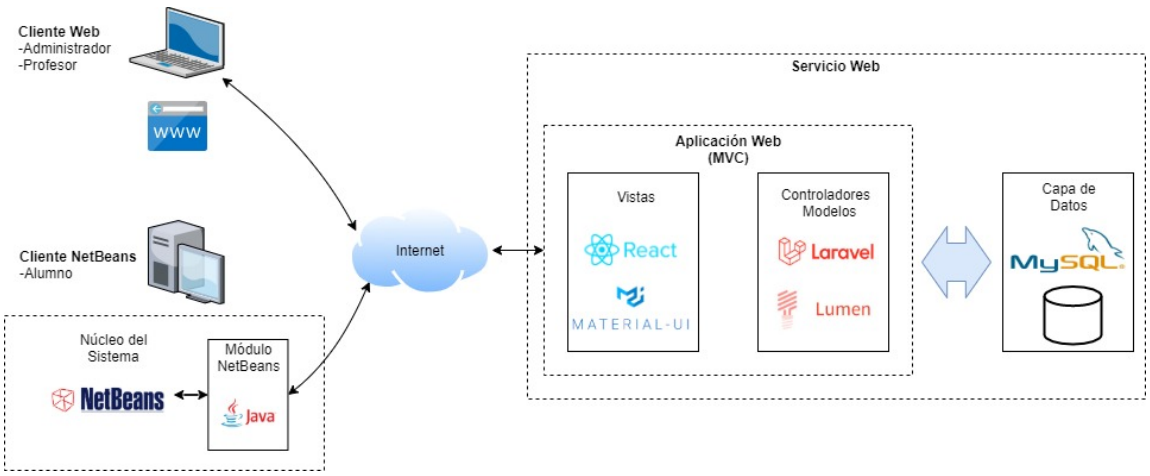


Figura 4.11: Representación de la Arquitectura Física de la herramienta.

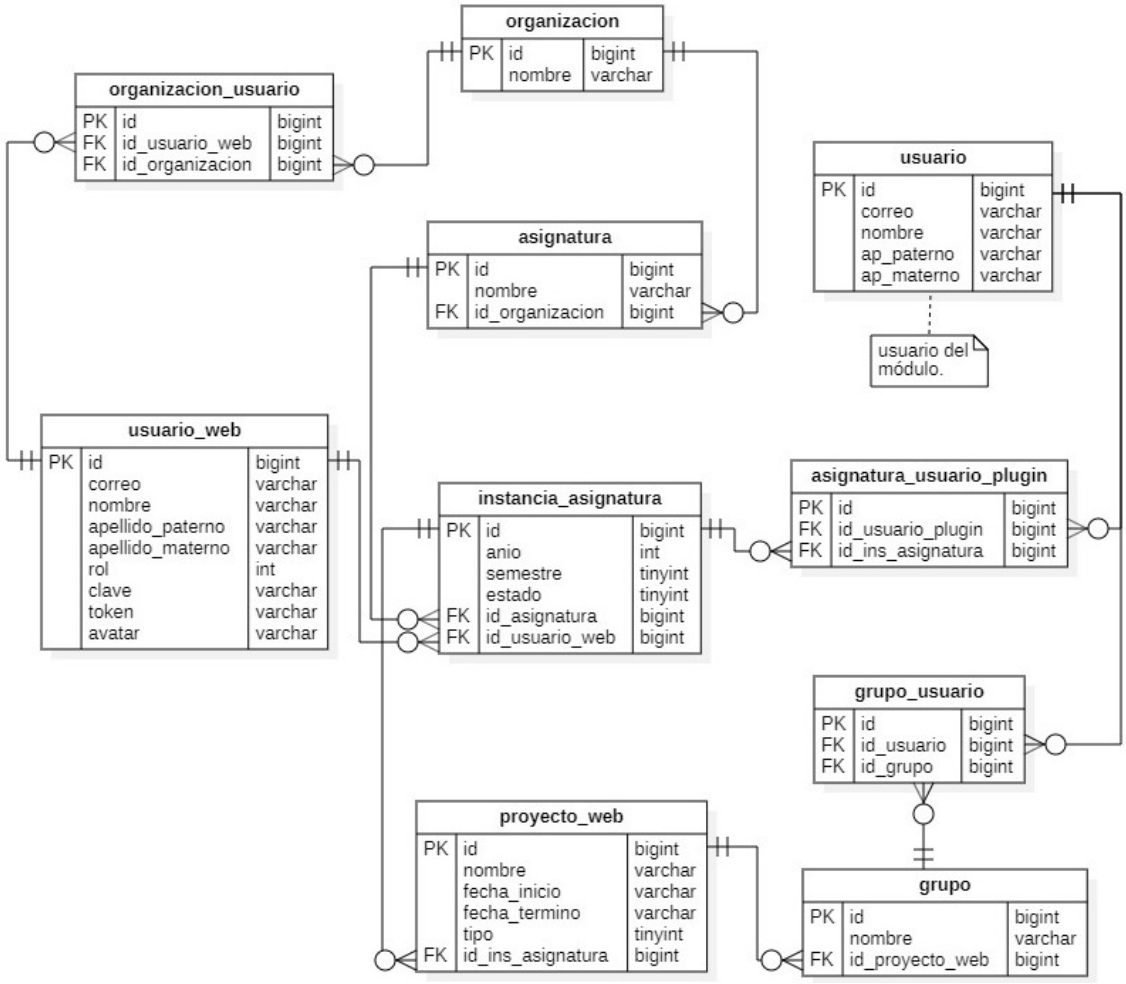


Figura 4.12: Modelo de datos para la aplicación web.

Estado del prototipo: Se define el modelo de datos completo para la herramienta propuesta, este se encuentra disponible en el **Anexo A.3**. Además, se implementa la comunicación entre la aplicación React y la *API REST* de Lumen delimitando la arquitectura física y lógica de la herramienta propuesta.

4.3.7. Iteración 7

El objetivo de esta iteración es definir las vistas que usa el rol de Administrador en la aplicación web. Las historias de usuario involucradas son **E011, E012, E017** Y **E018**.

Se comienza con la implementación de la vista de Organizaciones (Figura 4.13)

donde se encuentran las instituciones de educación superior como por ejemplo Universidades o Institutos. Luego, se trabaja en la vista de Usuarios de la Aplicación Bluelogger (Figura 4.14) donde es posible definir 2 tipos de roles, el Administrador o el Profesor. Por último, se desarrolla la vista de Módulos (Figura 4.15) útil para crear las asignaturas de programación relacionadas a cada Organización. Cada una de estas vistas son usadas por el rol de Administrador el cual tiene la facultad de listar, crear y editar los registros de cada una de ellas. Además, para listar los registros se generaliza la lógica de búsqueda, paginación y ordenamiento.

También, se definen los permisos a las distintas vistas que tiene la aplicación web según el tipo de usuario que usa la aplicación. Por ejemplo, al listado de Organizaciones puede acceder el Administrador y el Profesor con la diferencia de que sólo el Administrador tiene acceso a las vistas para crear y editar Organizaciones.

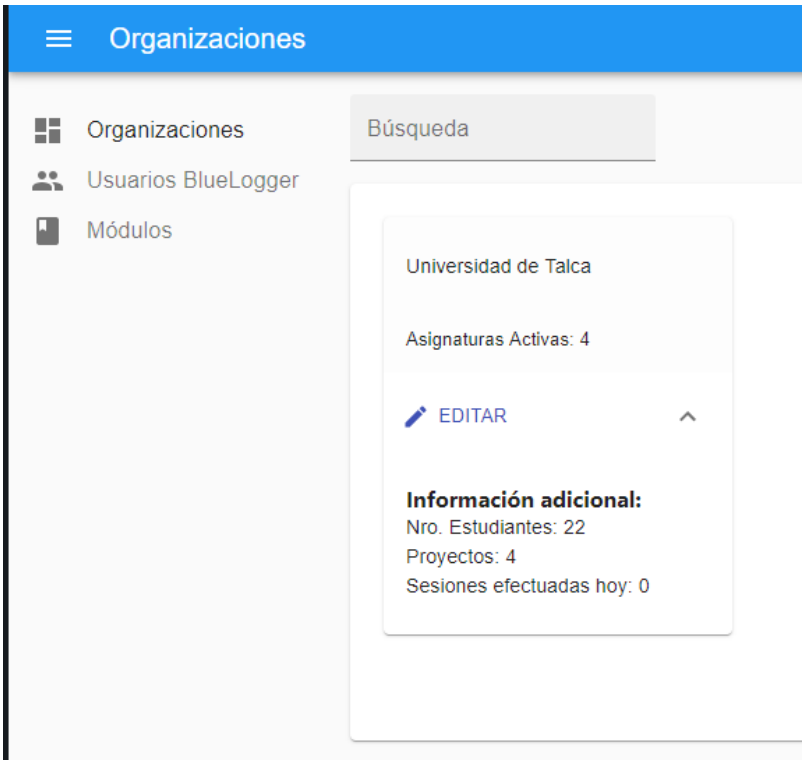


Figura 4.13: Vista de Organizaciones para el Administrador.

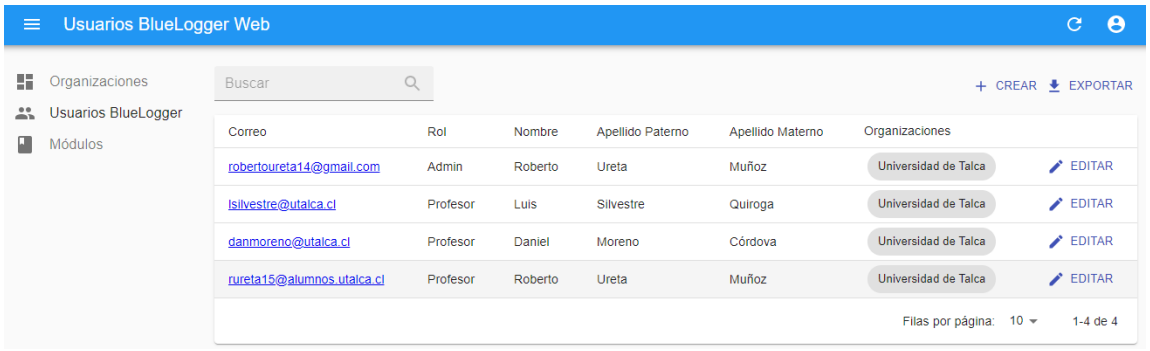


Figura 4.14: Vista de Usuarios de BlueLogger para el Administrador.

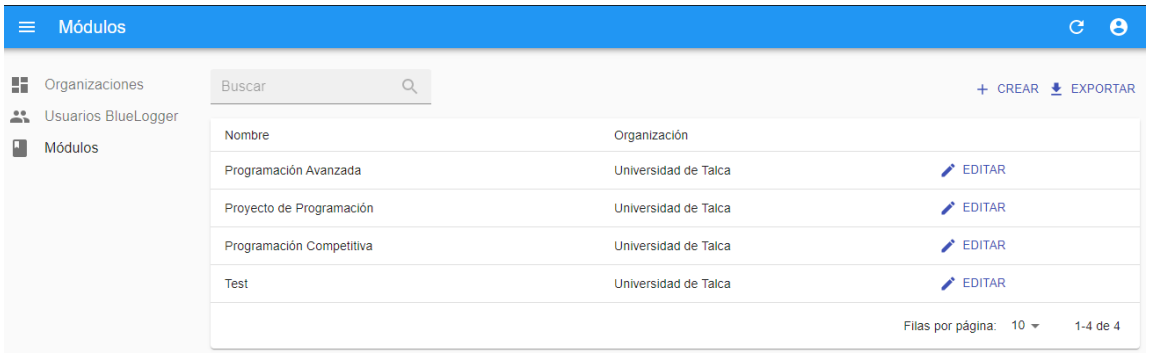


Figura 4.15: Vista de los Módulos para el Administrador.

Estado del prototipo: Las vistas usadas por el Administrador se encuentran completamente funcionales. Además, para acceder a las distintas vistas y rutas el usuario conectado debe estar autorizado, de otra manera, no se puede acceder al recurso. Todas las vistas relacionadas con el usuario del tipo Administrador se encuentran disponibles en el **Anexo A.4**.

4.3.8. Iteración 8

El objetivo de esta iteración es comenzar con la definición de las vistas utilizadas por el rol de Profesor de la aplicación web. Las historias involucradas son **E012, E013** y **E014**.

Primero, cuando el usuario que accede a la aplicación es un Profesor, en la vista principal solo aparecen su organizaciones asociadas, al presionar sobre una de estas tarjetas se carga la vista de Cursos (Figura 4.16). Aquí el Profesor tiene la facultad

de crear instancias de los módulos registrados por el Administrador según el año y semestre en que se cursa cierto módulo.

Además, se agregan dos botones, por cada curso, útiles para acceder a las vistas de Proyectos y Estudiantes asociados. En la primera (Figura 4.17) se muestran los proyectos individuales o grupales de cada curso. Por cada proyecto se genera un código que será el nombre del proyecto en NetBeans en el que trabajaran los alumnos. Para el caso de la vista de Estudiantes asociados, en esta iteración se implementan 2 formas de agregar alumnos a un curso, la primera consiste en registrar al estudiante desde cero (Figura 4.18), la segunda es mediante un campo de búsqueda por correo para los estudiantes que ya están registrados en la base de datos de la herramienta. En ambos componentes es posible listar, crear y editar registros.

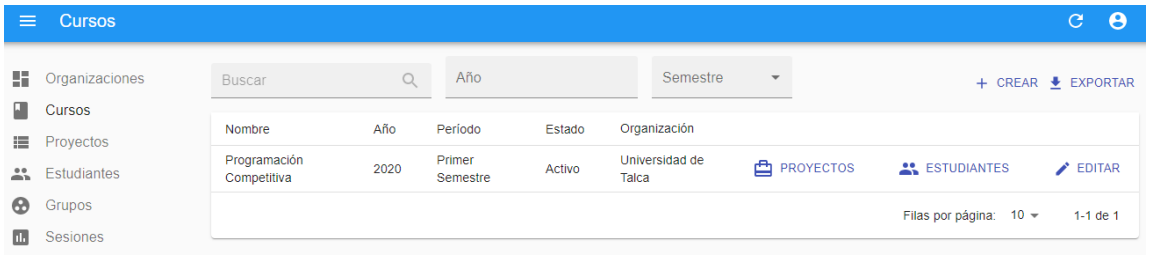


Figura 4.16: Vista de Cursos de un Profesor.

Figura 4.17: Vista para listar proyectos asociados a un curso.

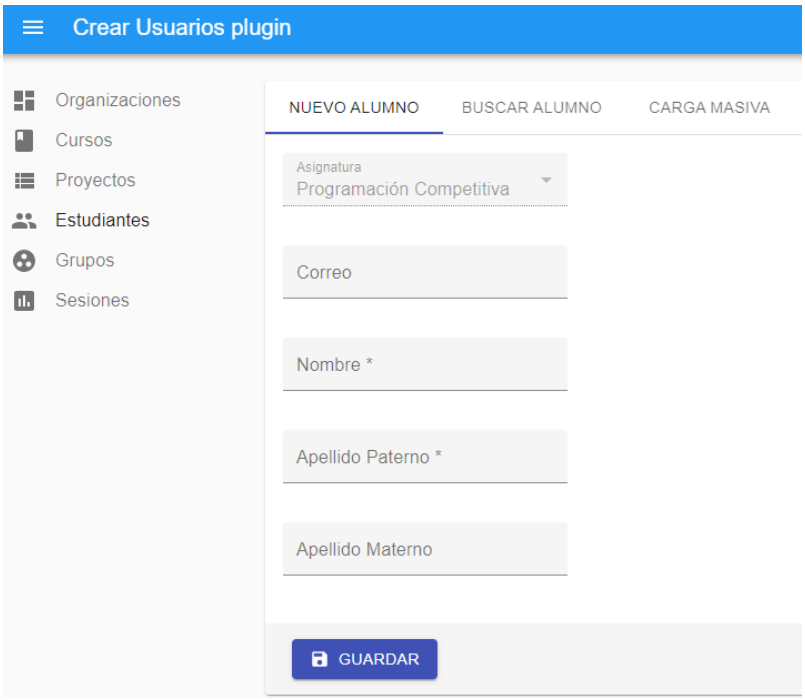


Figura 4.18: Vista para agregar estudiantes a un curso.

Estado del prototipo: Las vistas de Cursos, Proyectos y Estudiantes usadas por el Profesor se encuentran operativas, siendo posible la gestión de estudiantes y proyectos individuales o grupales asociados a cierto curso. Las demás vistas implementadas en esta iteración que se relacionan con el usuario del tipo Profesor se encuentran disponibles en el **Anexo A.5**.

4.3.9. Iteración 9

El objetivo de está iteración es finalizar con la definición de las vistas usadas por el Profesor. Entre estas vistas se incluye la de Sesiones donde se encuentran las métricas de desarrollo. Las historias de usuario involucradas son **E014** y **E015**.

Para comenzar, se agrega el método de carga masiva de alumnos a un curso. Esto se lleva a cabo usando el documento con formato *.xlsx* el cual contiene a los participantes del curso y está disponible en Educandus. También, se agregan las vistas que permiten listar, crear y editar grupos asociados a un proyecto determinado.

Por último, se agrega la vista de sesiones asociadas a un estudiante o a un grupo según sea el caso. En esta vista es posible listar las distintas sesiones generadas por

los estudiantes (Figura 4.19). Además, cada registro se puede expandir para mostrar el detalle de la sesión en particular, en este apartado se agregan las métricas relacionadas con el tiempo de la sesión (Figura 4.21), métricas relacionadas con las clases de cierto proyecto, métricas relacionadas con los métodos de las clases visualizadas (Figura 4.20) y por último, métricas relacionadas con eventos de inserción o eliminación producidos durante la sesión.

Figura 4.19: Vista para visualizar sesiones de un estudiante.

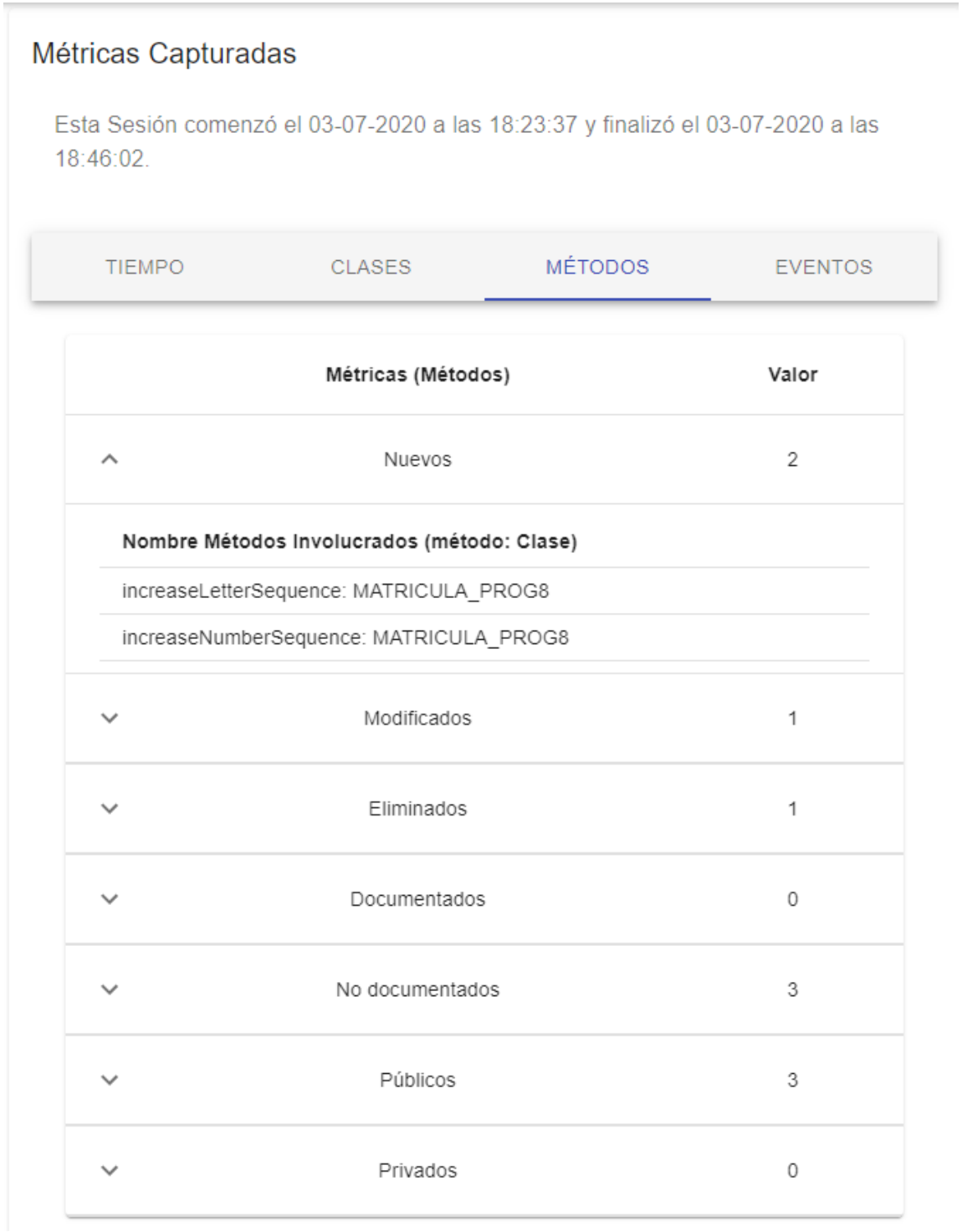


Figura 4.20: Detalle de la sesión de un estudiante (Métodos).

Figura 4.21: Detalle de la sesión de un estudiante (Tiempo).

Estado del prototipo: En este punto, todas las vistas de la aplicación web usadas por el Profesor se encuentran funcionales. Ya es posible visualizar el detalle de las distintas sesiones realizadas por los estudiantes al ocupar el IDE NetBeans. Todas las vistas implementadas en esta iteración están disponible en el **Anexo A.6**.

4.3.10. Iteración 10

El objetivo de esta iteración es agregar detalles menores a la aplicación web, para posteriormente desplegar la herramienta BlueLogger en su totalidad.

Para comenzar, en el apartado de detalle de una Sesión se agrega la opción de conocer el nombre de las clases o métodos involucrados en las métricas mostradas, esto se añade en el apartado de Clases, Métodos y Eventos como lo muestra la Figura 4.22.

Por último, para finalizar con el desarrollo de la herramienta BlueLogger, se compila y crea el archivo *.nbm* que contiene el módulo creado para la versión 8.2 del

IDE NetBeans y se despliega la aplicación web en el servicio de *hosting DigitalOcean*[3].

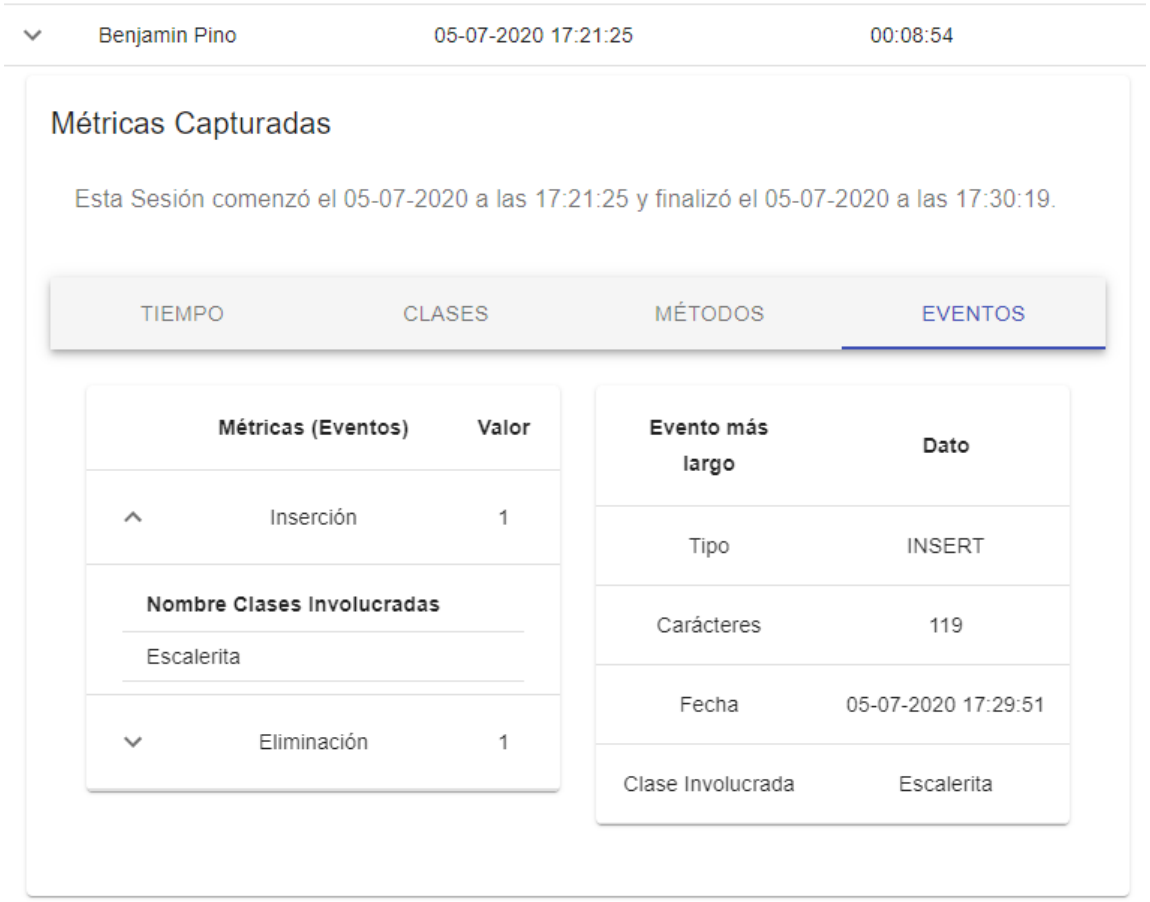


Figura 4.22: Nombre de Clases o Métodos involucrados en alguna métrica.

Estado final del prototipo: Esta es la última iteración según la planificación del proyecto. En este punto la herramienta BlueLogger ya se encuentra disponible para ser utilizada por estudiantes y profesores. Cabe destacar que, todas las vistas que no se muestran en este capítulo están disponibles en el **Anexo A** divididas de igual manera por iteraciones.

[3] `https://www.digitalocean.com`

5. Evaluación de BlueLogger

El objetivo de este capítulo es describir la evaluación de la herramienta propuesta. Para comenzar, se detalla la forma en que se aplica la metodología de evaluación seleccionada considerando cada una de sus etapas. Luego, se dan a conocer los resultados obtenidos seguido de su correspondiente análisis.

5.1. Fases de Experimentación

A continuación se detalla la realización de cada una de las fases de la experimentación en Ingeniería de Software según lo dispuesto en el Marco Teórico acotándolo a la evaluación de la herramienta BlueLogger.

5.1.1. Definición

Para llevar a cabo la etapa de Definición se utiliza el método GQM (*Goal-Question-Metric*) útil para definir diferentes aspectos de la evaluación [19], a continuación se destaca cada uno de estos aspectos a través de la definición de este estudio:

El **objetivo del estudio** es comprobar la instalación del módulo desarrollado y visualizar las distintas métricas de programación del módulo mediante la aplicación web. El **propósito** es validar el funcionamiento de la herramienta desarrollada al utilizar las interacciones capturadas de los distintos proyectos de los estudiantes. Esto se lleva a cabo desde la **perspectiva** del autor y del profesor guía en el **contexto** de clases virtuales dictadas a un curso ficticio de Programación Competitiva.

5.1.2. Diseño de la Experimentación

En esta etapa se definen los objetivos y características a evaluar junto con el instrumento usado para este propósito.

Los sujetos evaluados son estudiantes voluntarios pertenecientes al ultimo año de la carrera de Ingeniería Civil en Computación de la Universidad de Talca. Los estudiantes participan en un curso ficticio llamado Programación Competitiva creado para el propósito de esta experimentación. El encargado del curso es el autor del proyecto BlueLogger.

Objetivos de evaluación

Los objetivos a evaluar guardan relación con la definición entregada en la sección de Objetivos del Capítulo 1.

1. Evaluar si los estudiantes de un curso de programación pueden instalar el módulo en el IDE NetBeans.

2. Evaluar si los estudiantes de un curso de programación pueden usar el IDE NetBeans en conjunto con el módulo.

Características de evaluación

Las características a evaluar son aplicables a la utilización del módulo por parte de los sujetos voluntarios. De esta manera, para los estudiantes se estudia la Funcionalidad y Usabilidad del módulo. A continuación, se define cada una de estas características:

1. **Funcionalidad:** Capacidad de la herramienta desarrollada para proporcionar funciones que satisfacen las necesidades declaradas [33].

2. **Usabilidad:** Es la capacidad de la herramienta desarrollada para ser entendida, aprendida, usada y resulta atractiva para el usuario, cuando se usa bajo determinadas condiciones [33].

Instrumentos de Medición

Para cumplir con los objetivos de evaluación definidos se utiliza una Encuesta como instrumento de medición. La encuesta permite reunir información para realizar

un posterior análisis de los datos. Para construir la encuesta, se diseñan sentencias divididas por característica de evaluación usando la escala Likert [34].

5.1.3. Ejecución de la Experimentación

Para la ejecución del experimento se requiere estipular ciertas tareas a realizar por cada sujeto. Para esto se estipula un protocolo de experimentación acotando la ejecución de este proceso en las sesiones de experimentación.

Protocolo de Experimentación

Se define un protocolo para estipular las actividades que debe realizar cada sujeto que se somete a este experimento. Este procedimiento esta destinado a los estudiantes que utilizaran el módulo creado. El protocolo consiste en lo siguiente:

1. **Protocolo para estudiantes**
    - Presentación del módulo BlueLogger para IDE NetBeans.
    - Capacitación y consideraciones para la instalación del módulo.
    - Entrega de instructivo de instalación del módulo (disponible en el **Anexo B**).
    - Entrega de los proyectos en que deben trabajar.
    - Entrega de los nombres para cada proyecto, el cual deben utilizar a la hora de crear los proyectos en IDE NetBeans.

Sesiones de Experimentación

Para llevar a cabo el experimento se realiza la evaluación con un curso en que se trabajara en dos proyectos individuales usando el IDE NetBeans en su versión 8.2.

El grupo corresponde a seis estudiantes que participan del curso ficticio de Programación Competitiva. La sesión tiene una duración de 3 días consecutivos con periodos de programación que pueden oscilar entre 15 a 90 minutos por día.

Los usuarios a cargo de visualizar las información obtenida a través de la aplicación web serán el autor y el profesor guía de este proyecto. Esto se debe principalmente a la modalidad online del semestre en que se realiza este trabajo y la dificultad que se tiene a la hora de contactar a un profesor voluntario.

5.1.4. Análisis de la Experimentación

A continuación, se visualiza el análisis de los resultados obtenidos de la encuesta aplicada a los estudiantes voluntarios. Este análisis se divide según las secciones contempladas en la encuesta disponible en el **Anexo C**.

Resultados de Estudiantes

La primera característica evaluada es la de Funcionalidad del módulo BlueLogger. Los resultados se aprecian en el gráfico de la Figura 5.1 y en la Tabla 5.1.

De acuerdo a los datos recopilados es posible señalar que un 97 % de las respuestas de esta sección son positivas a la hora de probar la Funcionalidad del módulo, es decir que, durante el proceso de validación que sigue el estudiante no existieron mayores inconvenientes al momento de por ejemplo, ingresar el correo institucional, recibir e ingresar el código de validación.

La sentencia "*El módulo envía el código de validación al correo*", cuenta con una respuesta distinta a las demás, correspondiente al 3 % del total de datos para esta sección. Es posible que dicha respuesta guarde relación con uno de los sujetos que participó en el experimento, el estudiante usaba un antivirus que bloqueaba el envió del correo electrónico con el código. Con lo cual, en un principio no podía activar el módulo.

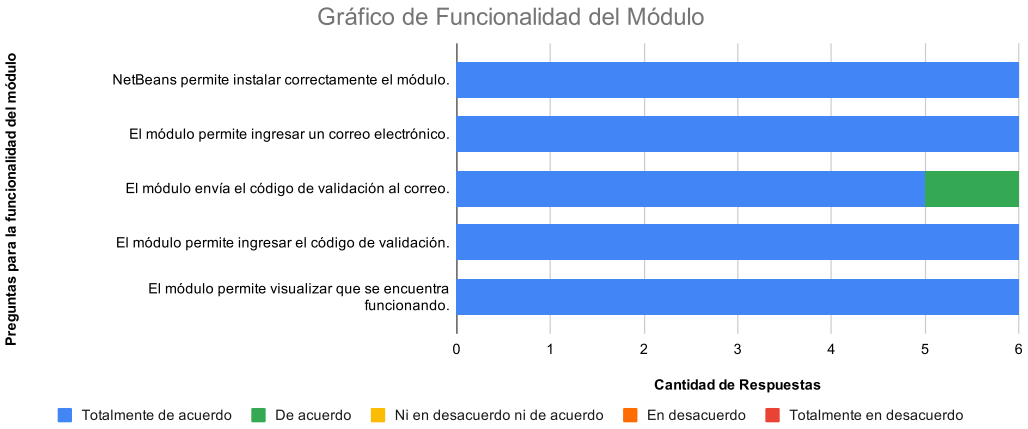


Figura 5.1: Resultado de evaluación de la característica de Funcionalidad para el módulo.

Tabla 5.1: Datos de la Funcionalidad del módulo BlueLogger.

	Totalmente de acuerdo	De acuerdo	Ni en desacuerdo ni de acuerdo	En desacuerdo	Totalmente en desacuerdo
NetBeans permite instalar correctamente el módulo.	6	0	0	0	0
El módulo permite ingresar un correo electrónico.	6	0	0	0	0
El módulo envía el código de validación al correo.	5	1	0	0	0
El módulo permite ingresar el código de validación.	6	0	0	0	0
El módulo permite visualizar que se encuentra funcionando.	6	0	0	0	0
% de la sumatoria del número de respuestas	**97 %**	**3 %**	**0 %**	**0 %**	**0 %**

La segunda característica evaluada es la de Usabilidad del módulo BlueLogger. Los resultados se aprecian en el gráfico de la Figura 5.2 y en la Tabla 5.2.

De acuerdo a los datos recopilados las respuestas para esta sección se agrupan entre *Totalmente de acuerdo* y *De acuerdo* con un 60 % y un 30 %, respectivamente. Es posible que esto se deba a que el módulo sólo interactúa con el estudiante la primera vez que se utiliza, al momento de solicitar el ingreso del correo y el código de validación. Luego, sólo se indica que el módulo esta funcionando.

Lo anterior, se ve reflejado en las sentencias de "*La interfaz del módulo es intuitiva"* y *"La cantidad de información entregada por el módulo es suficiente*", las cuales contienen la mayor diversidad de respuestas. Indicando, que los estudiantes no tienen suficiente conocimiento acerca del trabajo que realiza el módulo BlueLogger.

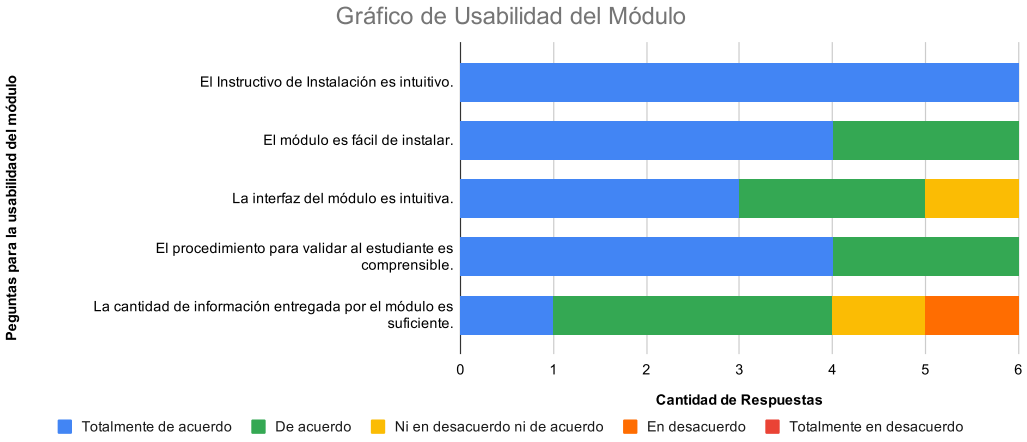


Figura 5.2: Resultado de evaluación de la característica de Usabilidad para el módulo.

Tabla 5.2: Datos de la Usabilidad del módulo BlueLogger.

	Totalmente de acuerdo	De acuerdo	Ni en desacuerdo ni de acuerdo	En desacuerdo	Totalmente en desacuerdo
El Instructivo de Instalación es intuitivo.	6	0	0	0	0
El módulo es fácil de instalar.	4	2	0	0	0
La interfaz del módulo es intuitiva.	3	2	1	0	0
El procedimiento para validar al estudiante es comprensible.	4	2	0	0	0
La cantidad de información entregada por el módulo es suficiente.	1	3	1	1	0
% de la sumatoria del número de respuestas	**60 %**	**30 %**	**7 %**	**3 %**	**0 %**

Para finalizar, se realizan dos preguntas con la finalidad de conocer apreciaciones generales de los sujetos encuestados. A continuación, se visualizan las preguntas seguido de sus respectivas respuestas.

1. ¿El módulo presentó algún error al momento de utilizarse?

    - Por mi parte no hubo ningún inconveniente.

- El plugin se instaló correctamente y no generó problema alguno durante su funcionamiento.
- no
- no
- Solamente una dificultad a la hora de generar el código en la instalación (que no llegaba al correo), se tuvo que desactivar el antivirus para poder generarlo.
- No hubieron errores en el momento durante la ejecución del mismo, sin embargo hubo un pequeño mensaje de incompatibilidad al iniciar NetBeans, no obstante no perjudicó su ejecución.

2. ¿Qué cambios o mejoras le haría usted al módulo?

- Quizás que muestre el tiempo de la sesión al estudiante que este usando el plugin.
- Me gustaría que tuviera un menú o la opción de escoger que proyectos son monitoreados, para que no haya flujo de información de proyectos privados que no deseo compartir.
- ninguno
- Mostrar algún resumen de las estadísticas recolectadas al usuario también o indicar donde pueden verlas. Debería haber un manual con respecto a que información esta recolectando por temas relacionados a la privacidad.
- Hace falta alguna interfaz que me permita conocer mis estadísticas. También que se entregue algún feedback para saber que está haciendo el plugin.
- Ampliar el modulo hacia otros IDE o editores de texto como Visual Studio Code.

Al analizar las respuestas de los estudiantes encuestados, se aprecia que en general no tuvieron problemas para instalar y activar el módulo en NetBeans, a excepción de un caso que tuvo que desactivar el antivirus para recibir el correo electrónico de validación.

Por otro lado, la mayoría de los estudiantes creen que el módulo debería mostrar la información que se esta recopilando, a través de alguna interfaz que muestre estadísticas o permita seleccionar los proyectos a monitorizar, por ejemplo.

5.1.5. Consideraciones adicionales de experimentación

La finalidad de esta sección es presentar evidencia del funcionamiento del módulo a través de algunos ejemplos capturados desde la aplicación web BlueLogger, demostrando de esta forma el correcto funcionamiento del módulo.

Para comenzar, en la Figura 5.3 se visualizan los dos proyectos que se desarrollan durante la experimentación.

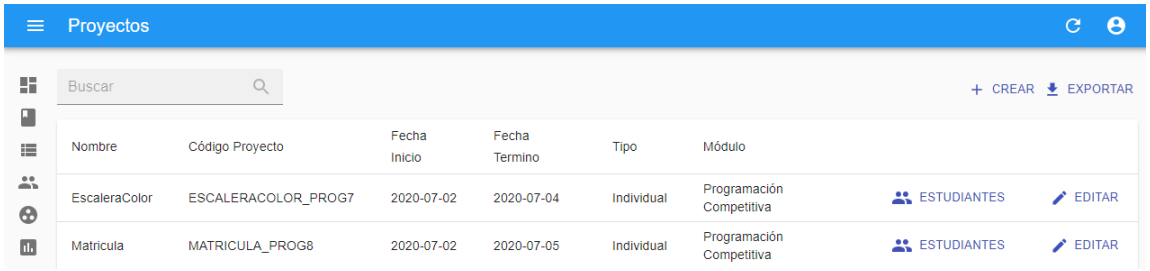


Figura 5.3: Proyectos realizados en la experimentación.

En la Figura 5.4 es posible visualizar a los seis estudiantes que registraron sesiones durante el periodo de experimentación.

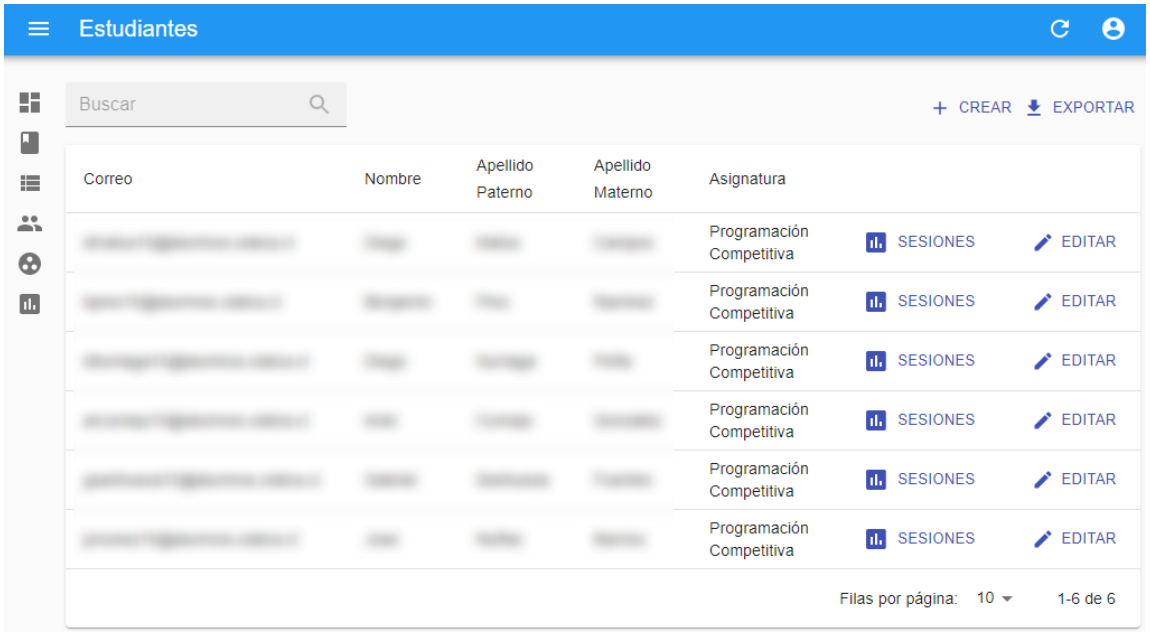


Figura 5.4: Estudiantes registrados en el curso ficticio de Programación Competitiva.

En la Figura 5.5 se visualiza las sesiones de un estudiante en específico, donde se

destaca las fechas en que se realiza cada sesión.

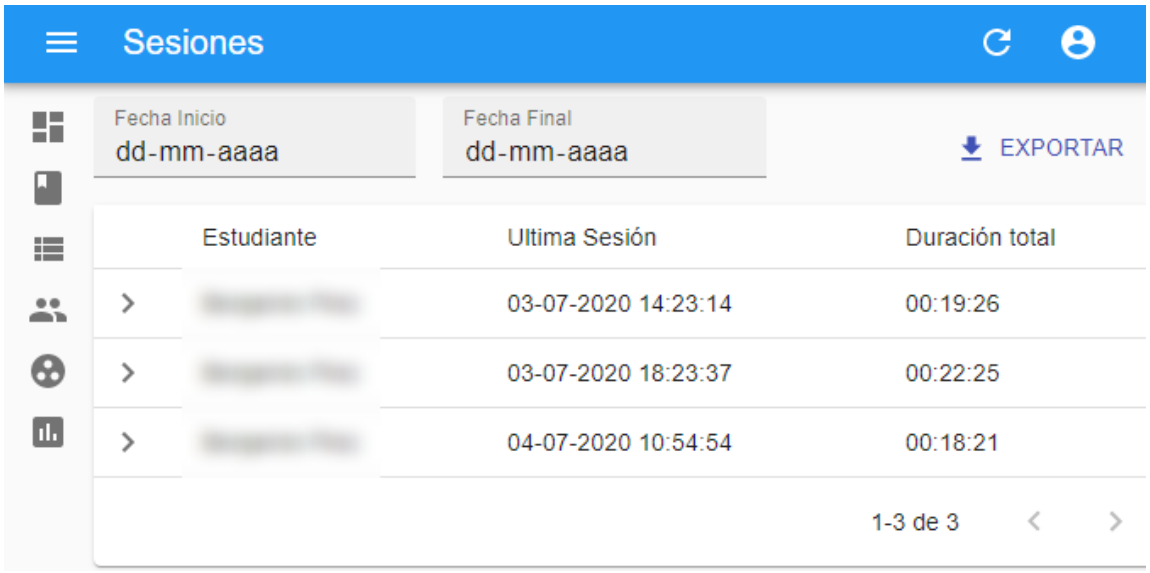


Figura 5.5: Sesiones realizadas por un estudiante.

Por último, en las Figuras 5.6 y 5.7 se aprecian dos ejemplos del detalle de una sesión efectuada por distintos estudiantes, visualizando específicamente las métricas de tiempo y métodos capturados.

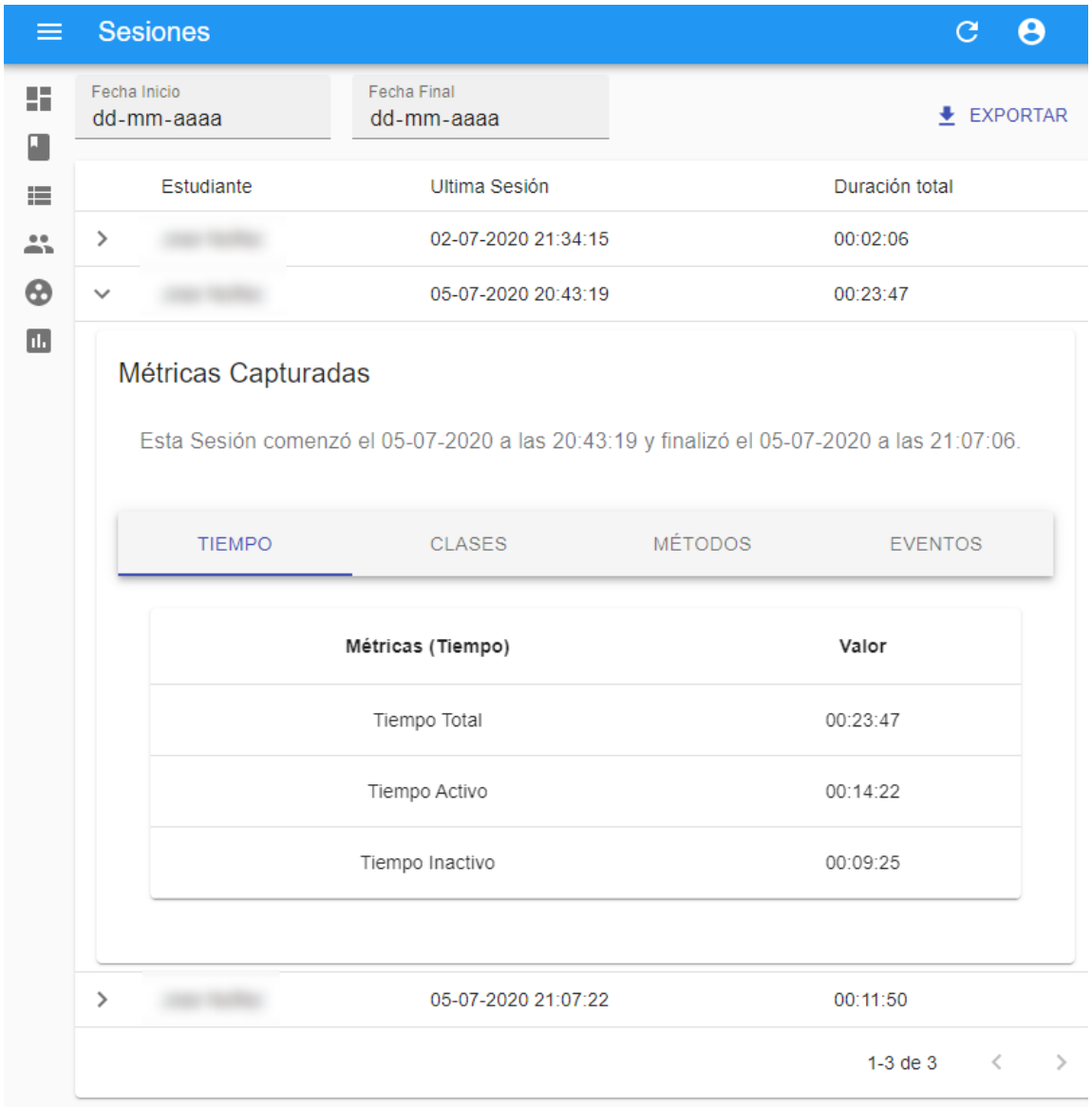


Figura 5.6: Métricas de tiempo obtenidas en la experimentación.

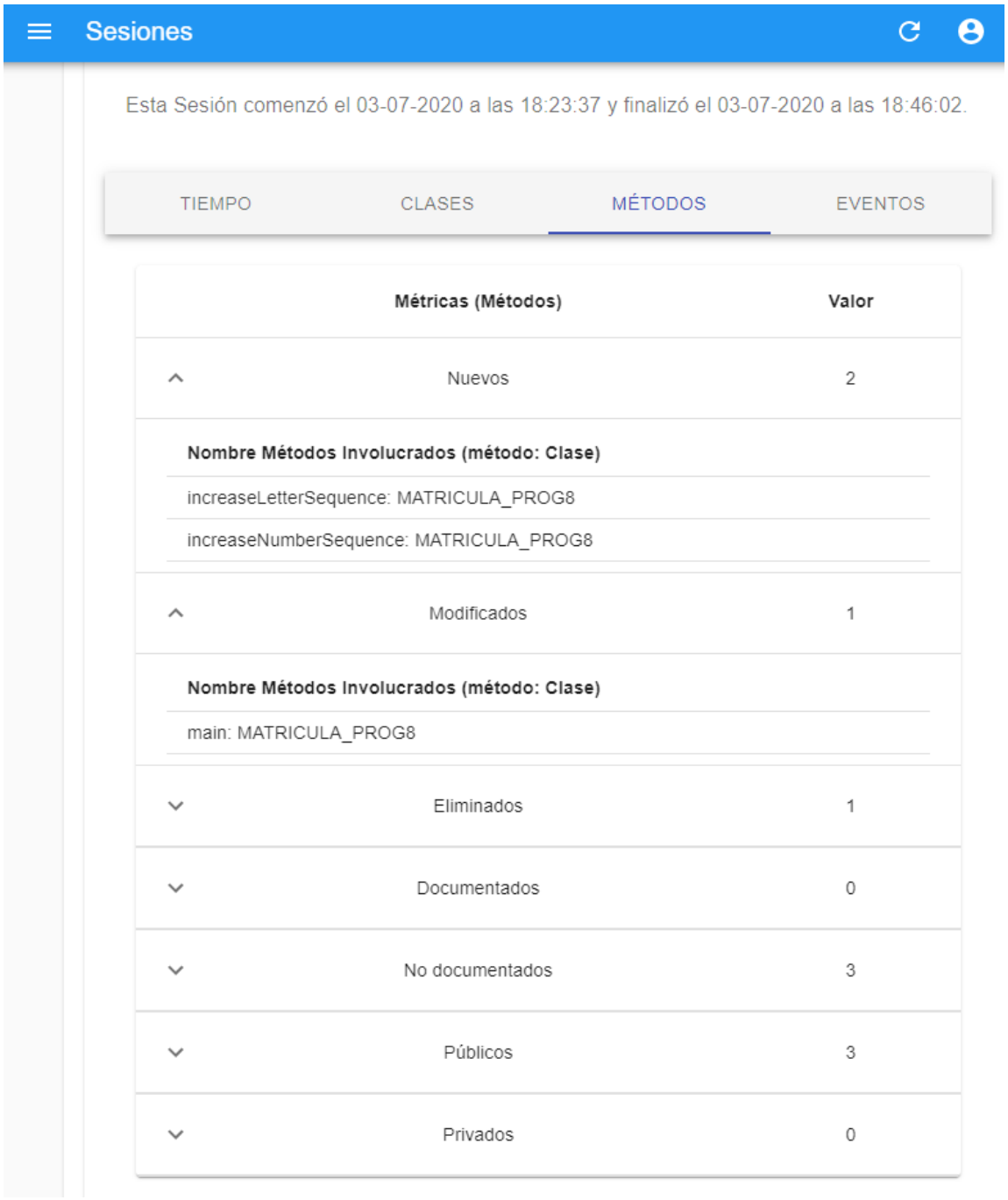


Figura 5.7: Métricas de métodos obtenidas en la experimentación.

5.2. Revisión de resultados de experimentación

En esta sección se hace un análisis relacionado con los objetivos de evaluación planteados previamente comparándolos con la encuesta realizada, entregando conclusiones generales de esta experimentación.

Es importante señalar que el total de estudiantes voluntarios lograron instalar el módulo de BlueLogger en el IDE NetBeans en su versión 8.2. Cumpliendo de esta forma lo estipulado en el primer objetivo de evaluación, el cual busca demostrar la validación y funcionamiento del módulo BlueLogger.

Por otro lado, el segundo objetivo a evaluar también se cumple, ya que, la aplicación web recibió distintas sesiones de todos los estudiantes evaluados, tal como se aprecia en el apartado de **Consideraciones adicionales de experimentación** presente en la Sección anterior. Con lo cual, los estudiantes no tuvieron inconvenientes a la hora de trabajar en los proyectos designados. Esto se demuestra, en las preguntas de la encuesta donde se capturan distintas apreciaciones de los estudiantes, en estas, no se reporta ningún error que suceda durante la utilización del IDE NetBeans.

6. Conclusiones y Trabajo Futuro

En este capítulo se precisan las conclusiones obtenidas durante el proceso de desarrollo del proyecto BlueLogger. A continuación, se expone el cumplimiento de los objetivos y la resolución sobre la efectividad del software construido. Además, se presentan las lecciones aprendidas a nivel tecnológico, metodológico y personal. Para finalizar, con el trabajo futuro que se desprende de este proyecto.

6.1. Conclusiones

Para comenzar, se recuerda los objetivos específicos de este proyecto indicando donde se cumple cada uno de ellos. El primero, es "*Caracterizar las interacciones genéricas que son aplicables en el desarrollo de un proyecto de software*", para el cual a través de información bibliográfica se observan las distintas interacciones de usuario detectables en el desarrollo de software quedando disponible a detalle al comienzo del **Capítulo 2** del presente documento proyecto.

El segundo objetivo es "*Seleccionar según las limitantes del Entorno de Desarrollo Integrado NetBeans las interacciones específicas a capturar*", el cual se realiza durante el desarrollo del módulo BlueLogger, específicamente en las iteraciones dos y tres disponibles en el **Capítulo 4** del presente documento.

El tercer objetivo contempla "*Desarrollar un módulo para el Entorno de Desarrollo Integrado NetBeans que sea capaz de capturar las interacciones de usuarios presentes en archivos de código fuente con extensión Java*". Esta tarea se lleva a cabo en las primeras 3 iteraciones del desarrollo expuestas a detalle en el **Capítulo 4**

del documento. Este fue uno de los objetivos más importantes y desafiantes de este proceso ya que es una pieza fundamental para solucionar el problema planteado.

El cuarto objetivo considera "*Definir las métricas de desarrollo específicas a utilizar en base a las interacciones capturadas*". Para esto, en primer lugar se define como funcionan las métricas en el desarrollo de software a través de información bibliográfica disponible a detalle en el **Capítulo 2**. Luego, en las iteraciones 8 y 9 del **Capítulo 4** se precisan las métricas específicas a utilizar en la aplicación web, esto en base a las interacciones capturadas por el módulo en NetBeans.

El quinto objetivo es "*Desarrollar una aplicación web que permita visualizar la información capturada por el módulo a través de métricas de desarrollo.*". Esto se realiza entre las iteraciones 5 y 9 del proceso de desarrollo de BlueLogger, detallado en el **Capítulo 4** del documento. La aplicación web resultante permite al profesor crear cursos y proyectos de trabajo, con la finalidad de visualizar las distintas sesiones de programación de los estudiantes, las cuales contienen las distintas métricas capturadas.

En general, gracias al cumplimiento de todos los objetivos específicos planteados y recordando el objetivo principal del proyecto el cual consiste en "*Desarrollar una herramienta que permita capturar las interacciones de un usuario en IDE NetBeans y muestre a través de una aplicación web la información recopilada usando métricas de desarrollo*". Es posible concluir que se cumple con las distintas funcionalidades requeridas por el cliente. Agregando un mecanismo adicional para la evaluación y análisis en torno al desempeño de los estudiantes. Siendo de utilidad para situaciones que requieran de evidencia para validar una opinión o decisión de los profesores frente al trabajo realizado por los estudiantes.

6.2. Lecciones Aprendidas

El desarrollo de este proyecto conllevó aprender un conjunto de nuevas lecciones, las cuales pueden ser aplicables en otros contextos de interés. A continuación, se mencionan cada una de estas enseñanzas.

6.2.1. A nivel tecnológico

Una de las elecciones más importantes fue ocupar la Arquitectura Micro-Kernel, ya que anteriormente no se tenía conocimiento de su implementación. Esta arqui-

tectura resulta interesante debido a su capacidad para incluir otro diseño aun más complejo.

Además, el proceso que conlleva desarrollar el módulo para NetBeans, deja nuevos aprendizajes sobre los *Listeners* implementados en el lenguaje Java y también la forma de analizar los archivos de origen para obtener información de las clases y métodos.

Por último, se destaca los nuevos aspectos aprendidos al usar *React* para crear la aplicación web. Diseñando componentes funcionales y reutilizables, facilitando el proceso de desarrollo para esta parte del proyecto.

6.2.2. A nivel metodológico

En este apartado, se destaca la importancia de seguir las distintas etapas impuestas por las metodologías de desarrollo y evaluación, ya que aseguran un resultado final robusto.

Por ejemplo, la decisión de usar PXP para el desarrollo del proyecto permitió enfocarse en las funcionalidades de la herramienta creada, dejando de lado artefactos o roles innecesarios cuando el trabajo es individual.

En el caso del proceso de evaluación, la metodología de experimentación en Ingeniería de Software permite definir objetivos de evaluación realistas. Además, sus distintas etapas permiten limitar correctamente el proceso para los sujetos evaluados definiendo sesiones y protocolos de experimentación. Por último, los resultados obtenidos son asertivos y es posible realizar un análisis coherente frente a los objetivos evaluados.

6.2.3. A nivel personal

En este aspecto, uno de los mayores desafíos presentes en este proyecto fue desarrollar el módulo para NetBeans, debido a que la documentación disponible para este tipo de desarrollo es escasa y obsoleta. Además, se debía capturar información de difícil acceso, como por ejemplo, la firma de los métodos codificados en archivos con extensión Java, descubriendo una librería de código abierto que ayudó en este propósito.

Por último, se destaca la importancia de las métricas capturadas y visualizadas a través de la aplicación web, ya que es información relevante que puede dar indicios

del comportamiento de cada estudiante a la hora de programar algún proyecto de software.

6.3. Trabajo Futuro

En relación al trabajo futuro que se desprende de este proyecto, es posible ampliar las funcionalidades y el contexto de la herramienta desarrollada. Incluyendo por ejemplo, más lenguajes de programación o más entornos de desarrollo.

Debido a las limitantes de NetBeans, existen algunas interacciones que no se logran capturar con este módulo. Por lo que, se podría replicar un módulo similar para IDEs como *Visual Studio Code*, que además se encargue de capturar interacciones de usuario relacionadas con ejecuciones de código, errores de sintaxis o refactorizaciones complejas.

Con las nuevas interacciones es posible agregar nuevas métricas presentes en el desarrollo de software. Además, el proyecto desarrollado no cuenta con un perfil para los estudiantes que utilizan el módulo, siendo interesante, crear un nuevo perfil en la aplicación web destinado a estos usuarios con la finalidad de mostrar nuevas métricas orientadas a mejorar el desempeño de los estudiantes en sus proyectos de software.

Por último, la aplicación web entrega la funcionalidad de exportar los distintos datos recopilados. Con esto, se da la oportunidad a los profesores de aplicar ciencia de datos sobre la información recopilada y, de esta forma obtener un análisis distinto o más complejo sobre el desempeño de los estudiantes a lo largo del desarrollo de un proyecto de software.

Bibliografía

[1] Mark Richards. *Software Architecture Patterns*. O'Reilly Media, Inc., 2017.

[2] J M Cepeda. Metodologia de la enseñanza basada en competencias. *Revista Iberoamericana de Educación*, 2005.

[3] Lázaro Bustio Martínez, Yenice Coma Peña, and Isneri Talavera Bustamante. Arquitectura basada en plugins para el desarrollo de software científico. 2013.

[4] Mauricio A. Saca. Refactoring improving the design of existing code. In *2017 IEEE 37th Central America and Panama Convention, CONCAPAN 2017*, page 9, 2018.

[5] WakaTime. WakaTime. `https://wakatime.com`, Consultado el: 27 de Agosto de 2019.

[6] Codealike. Codealike: Powerful Metrics for High Performance Developers. `https://codealike.com`, Consultado el: 27 de Agosto de 2019.

[7] Yue Liu. Visual Studio Code Coding Tracker. `https://github.com/hangxingliu/vscode-coding-tracker`, Consultado el: 27 de Agosto de 2019.

[8] Hubstaff. Hubstaff: Time Tracking and Productivity Monitoring Tool. `https://hubstaff.com`, Consultado el: 02 de Marzo de 2020.

[9] CodeMR. CodeMR Software Quality Tool. `https://www.codemr.co.uk/`, Consultado el: 31 de Agosto de 2019.

[10] Codetrails GmbH. Codecity. `https://marketplace.eclipse.org/content/codecity`, Consultado el: 31 de Agosto de 2019.

[11] Ideadapt. Metriculator. `https://github.com/ideadapt/metriculator`, Consultado el: 31 de Agosto de 2019.

[12] Eclipse. Trace Compass. `https://www.eclipse.org/tracecompass/`, Consultado el: 31 de Agosto de 2019.

[13] NetBeans. Simple Code Metrics. `http://plugins.netbeans.org/plugin/9494/simple-code-metrics`, Consultado el: 31 de Agosto de 2019.

[14] NetBeans. SourceCodeMetrics: Measures the Java source code metrics. `http://plugins.netbeans.org/plugin/42970/sourcecodemetrics`, Consultado el: 31 de Agosto de 2019.

[15] Romain Robbes and Michele Lanza. Characterizing and understanding development sessions. *IEEE International Conference on Program Comprehension*, pages 155–164, 2007.

[16] Romain Robbes. Mining a change-based software repository. *Proceedings - ICSE 2007 Workshops: Fourth International Workshop on Mining Software Repositories, MSR 2007*, 2007.

[17] Manu Pijierro. Métricas de calidad para componentes de software (packages), 2018. `https://medium.com/@mpijierro/metricas-de-calidad-para-componentes-de-software-packages-9d1280189d92`, Consultado el: 08 de Septiembre de 2020.

[18] Yani Dzhurov, Iva Krasteva, and Sylvia Ilieva. Personal Extreme Programming – An Agile Process for Autonomous Developers. (January 2009), 2014.

[19] Omar S. Gómez, Juan P. Ucán, and Gerzon E. Gómez. Aplicación del proceso de experimentación a la Ingeniería de Software. 8(2013):26–37, 2010.

[20] Abenza and Garrido Pablo. *Comenzando a programar con JAVA*. Universidad Miguel Hernández, 2015.

[21] NetBeans. NetBeans IDE. `https://netbeans.org/index_es.html`, Consultado el: 27 de Agosto de 2019.

[22] MDN. Acerca de JavaScript - JavaScript. `https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript#JavaScript_resources`, Consultado el: 11 de Marzo de 2020.

[23] Facebook Inc. React - Una biblioteca de JavaScript para construir interfaces de usuario. `https://es.reactjs.org`, Consultado el: 11 de Marzo de 2020.

[24] The PHP Group. PHP: ¿Qué es PHP? - Manual. `https://www.php.net/manual/es/intro-whatis.php`, Consultado el: 11 de Marzo de 2020.

[25] Taylor Otwell. Lumen - PHP Micro-Framework By Laravel. `https://lumen.laravel.com`, Consultado el: 09 de Marzo de 2020.

[26] Oracle Corporation. MySQL:: About MySQL. `https://www.mysql.com/about/`, Consultado el: 11 de Marzo de 2020.

[27] Stack Overflow. Stack Overflow, 2020. `https://insights.stackoverflow.com/survey/2020#technology`, Consultado el: 14 de Junio de 2020.

[28] Mike Cohn. *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[29] Max Rehkopf. ¿Qué es un tablero de kanban? `https://www.atlassian.com/es/agile/kanban/boards`, Consultado el: 21 de Junio de 2020.

[30] Postman. The Collaboration Platform for API Development. `https://www.postman.com/postman/`, Consultado el: 18 de Julio de 2020.

[31] Software Testing Fundamentals. Black Box Testing, 2020. `http://softwaretestingfundamentals.com/black-box-testing/`, Consultado el: 18 de Julio de 2020.

[32] Fred Swartz. Java Notes, 2010. `http://www.fredosaurus.com/notes-java/GUI/events/15listeners.html`, Consultado el: 29 de Junio de 2020.

[33] ISO 25000. ISO/IEC 25010, 2014. `https://iso25000.com/index.php/normas-iso-25000/iso-25010`, Consultado el: 13 de Julio de 2020.

[34] Manuel Guil Bozal. Escala mixta likert-thurstone. *ANDULI, Revista Andaluza de Ciencias Sociales*, (5):81–95, 2005.

ANEXOS

A. Documentación de las iteraciones

A continuación, se muestran los diagramas y Figuras que surgen en algunas de las iteraciones y por espacio no se agregan en el documento principal. Además, se deja a disposición del lector los enlaces a la Pila de Producto[1] y al tablero Trello[2] del proyecto.

A.1. Iteración 1

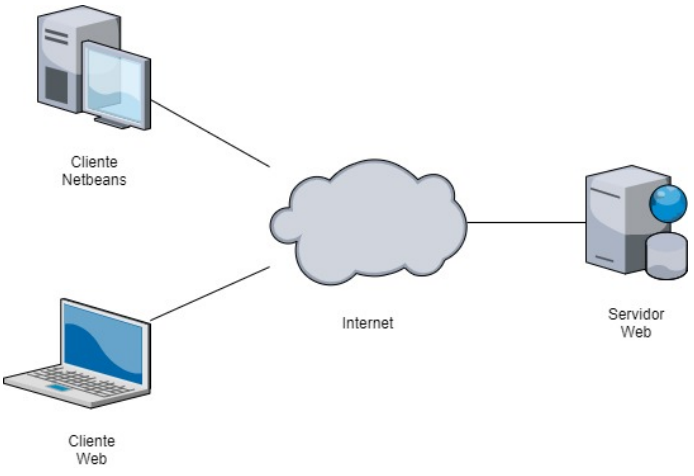


Figura A.1: Arquitectura Física de la herramienta BlueLogger.

[1] https://docs.google.com/spreadsheets/d/1U1xBHzWh6H92JQZVn4ZsyZsTTLnaaITylTeWVRJESAY/edit?usp=sharing

[2] https://trello.com/invite/b/yuqZVLWw/8ffdcafb07aa224a428e9b25e37f5a07/proyecto-de-titulo-roberto-ureta

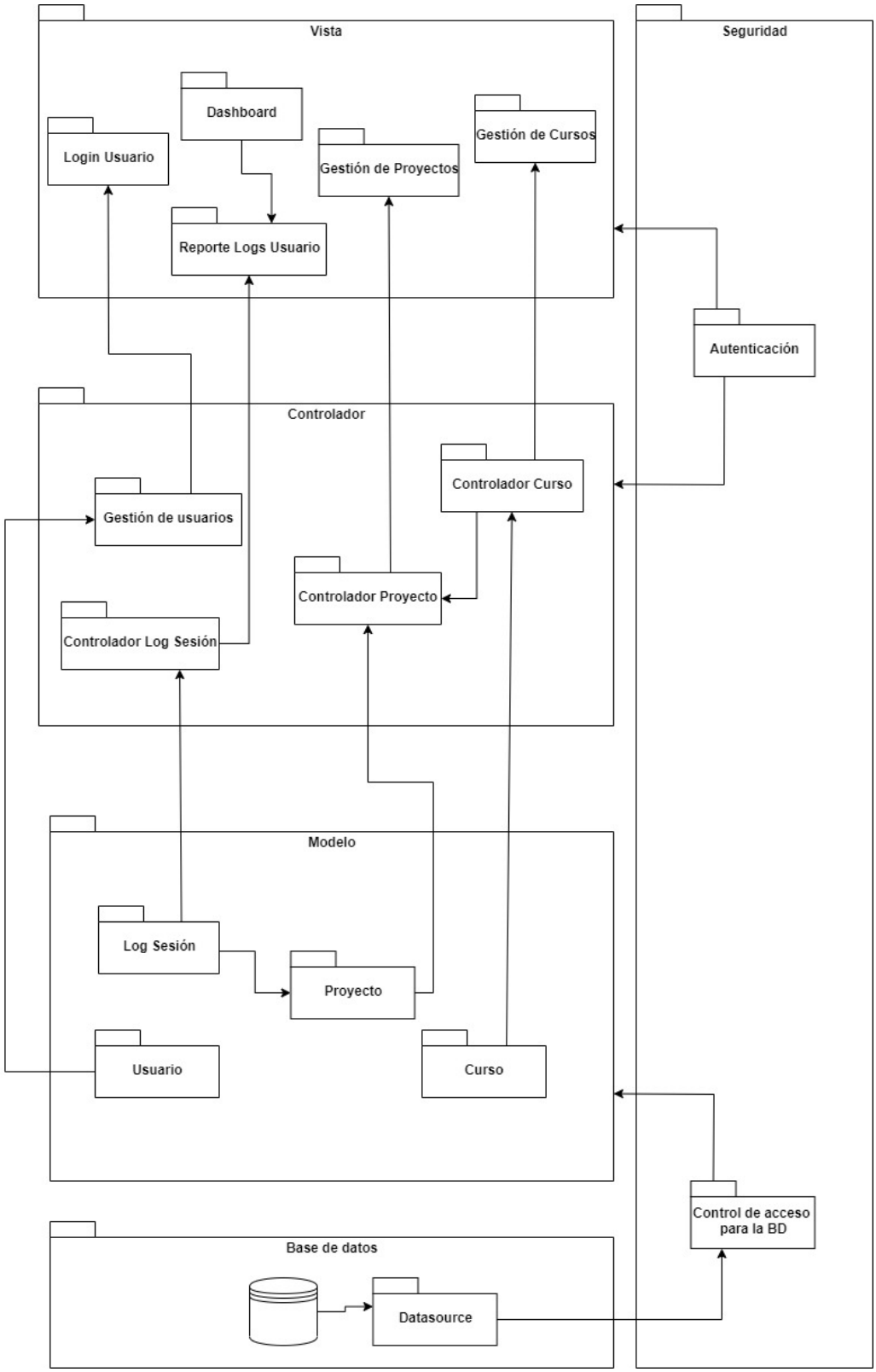


Figura A.2: Arquitectura Lógica Preliminar de la aplicación web.

A.2. Iteración 3

```
{
  "proyectos": [{
      "nombre": "nombreProyecto",
      "archivos": [{
          "nombreArchivo": "nombreArchivo1",
          "eventos": [...],
          "metodos": [...],
          "paquete": "paquete1",
          "cantLineasArc": 53,
          "segundosActivo": 87
        },
        {
          "nombreArchivo": "nombreArchivo2",
          "eventos": [...],
          "metodos": [...],
          "paquete": "paquete1",
          "cantLineasArc": 33,
          "segundosActivo": 58
        }
      ]
    }
  ],
  "usuario": "robertoureta14@gmail.com",
  "fechaInicio": "19/03/2020",
  "horaInicio": "19:15:37",
  "fechaFin": "19/03/2020",
  "horaFin": "19:17:58"
}
```

Figura A.3: Estructura completa del JSON enviado al servidor.

A.3. Iteración 6

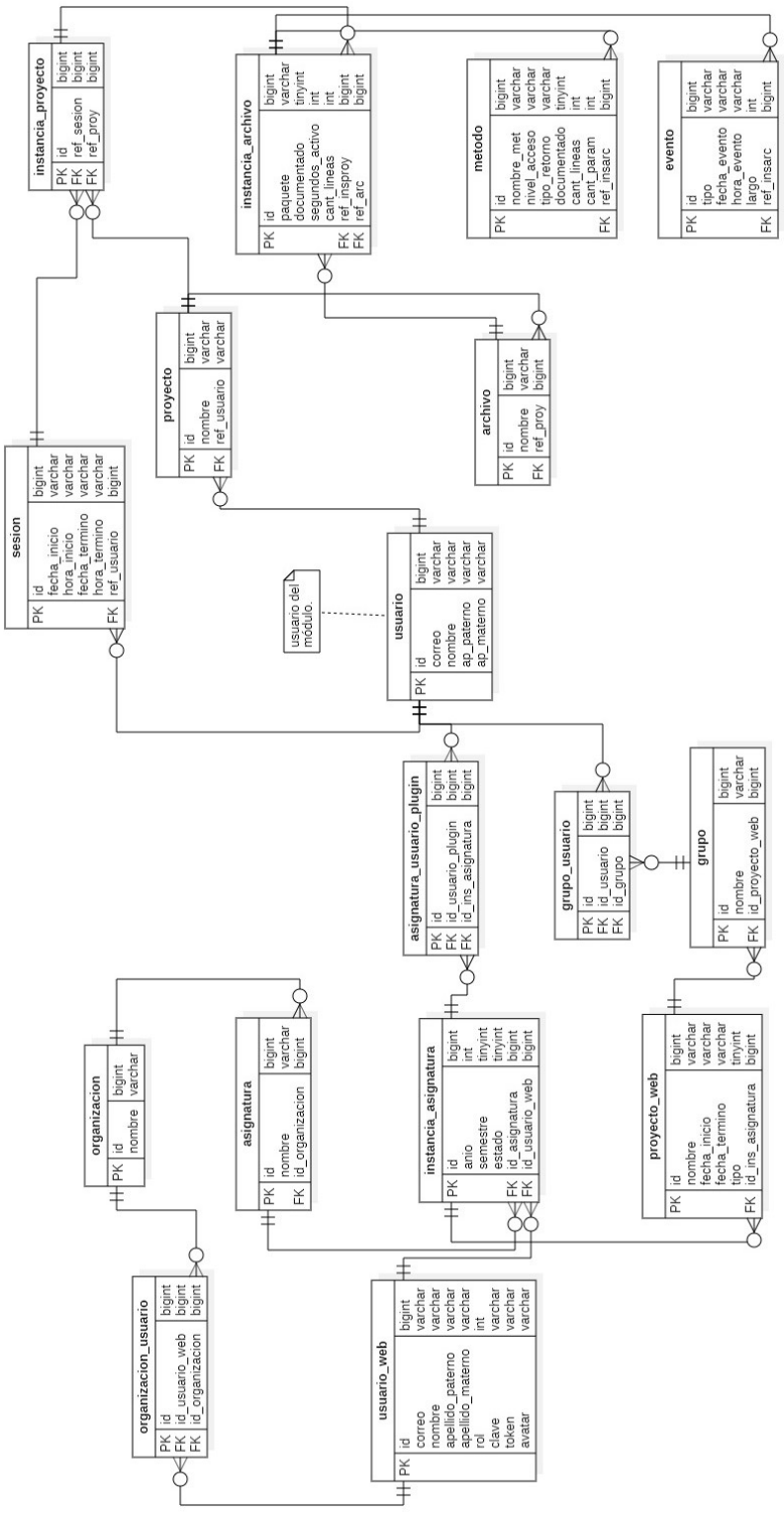


Figura A.4: Modelo de datos completo de la herramienta BlueLogger.

A.4. Iteración 7

Figura A.5: Login de la aplicación web BlueLogger.

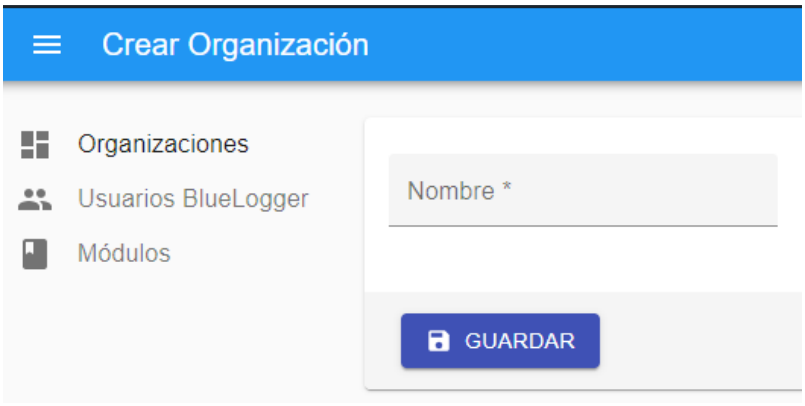


Figura A.6: Vista de Administrador para crear una organización.

Figura A.7: Vista de Administrador para editar una organización.

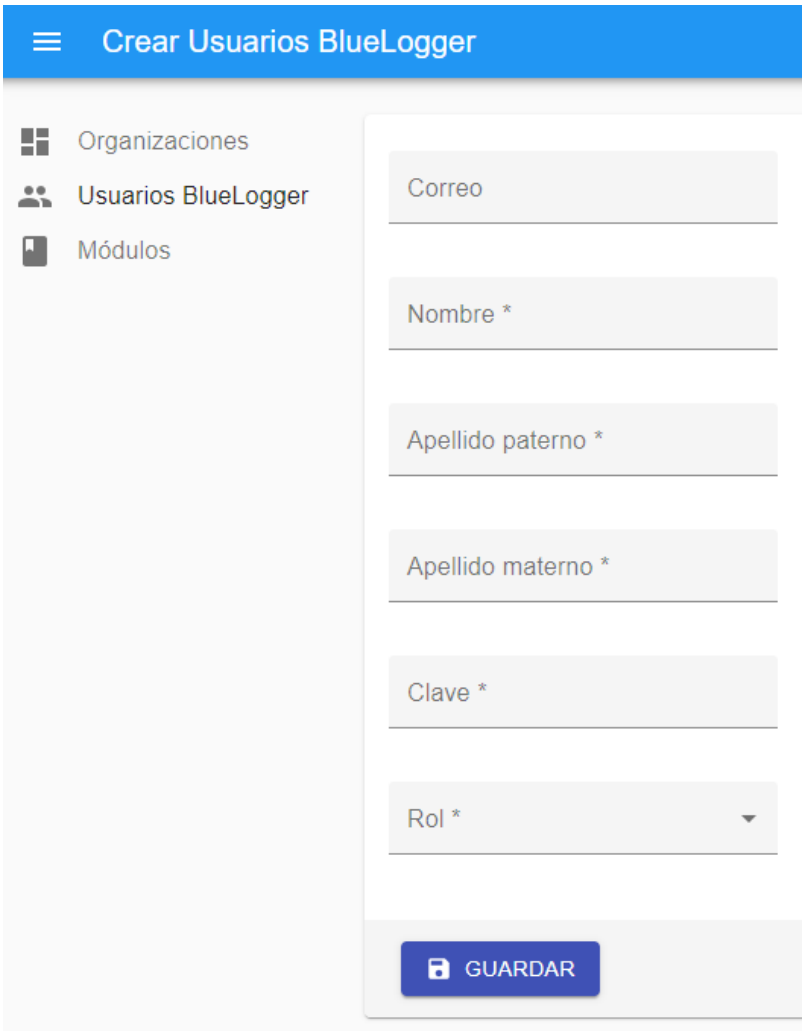


Figura A.8: Vista de Administrador para crear un usuario de la aplicación web BlueLogger.

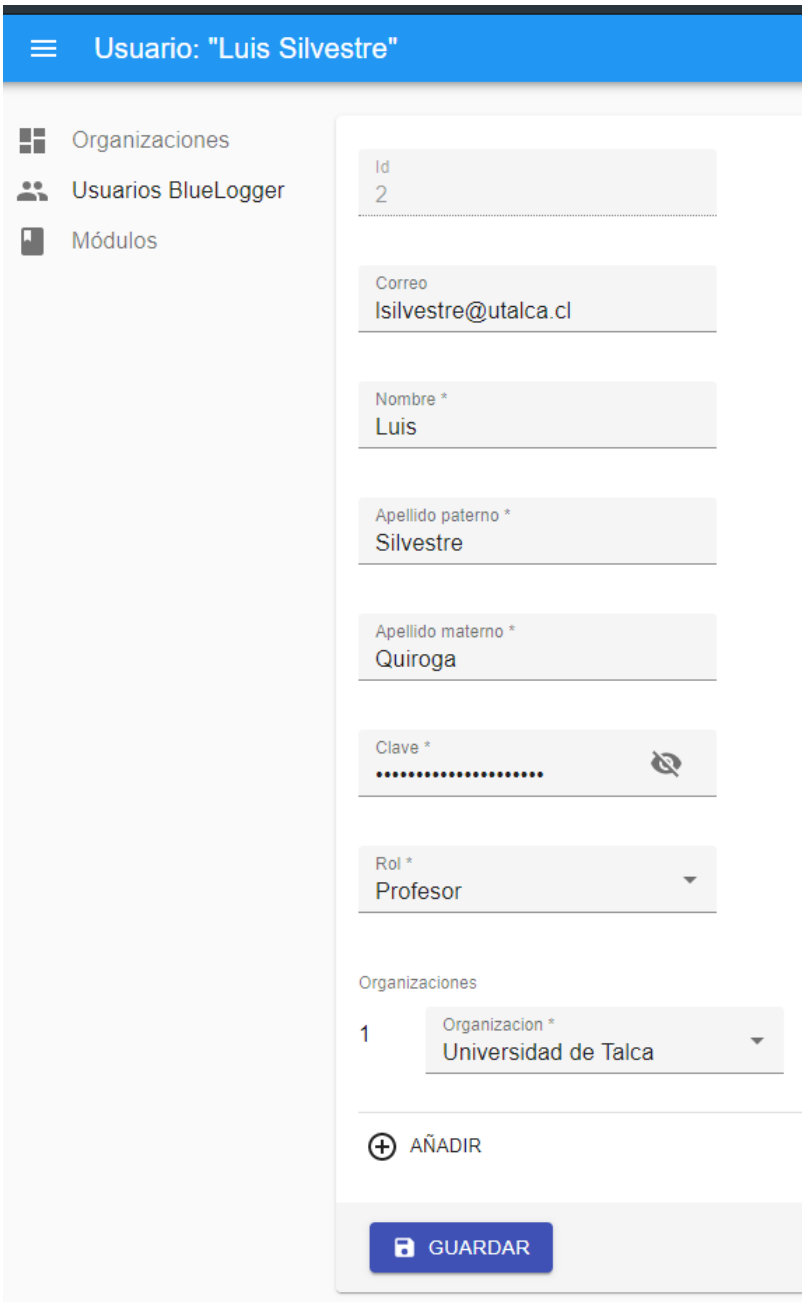


Figura A.9: Vista de Administrador para editar un usuario de la aplicación web BlueLogger.

Figura A.10: Vista de Administrador para crear un Módulo.

Figura A.11: Vista de Administrador para editar un Módulo.

A.5. Iteración 8

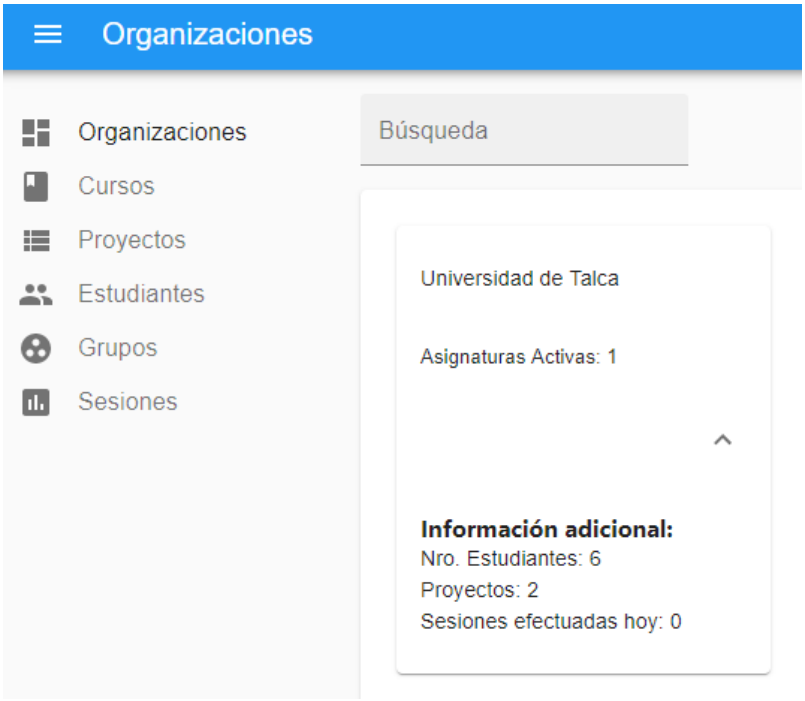


Figura A.12: Vista de Profesor para listar sus organizaciones asociadas.

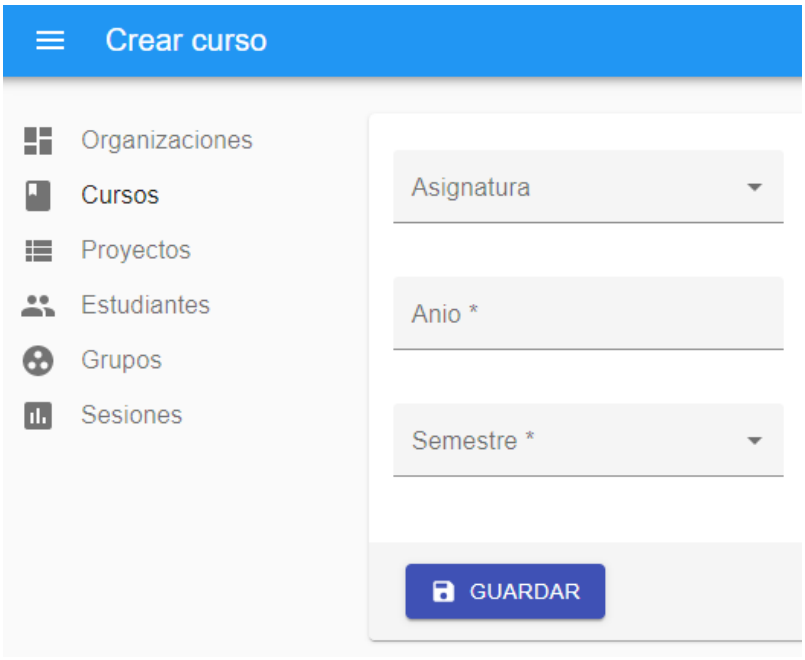


Figura A.13: Vista de Profesor para crear un Curso asociado a una Organización.

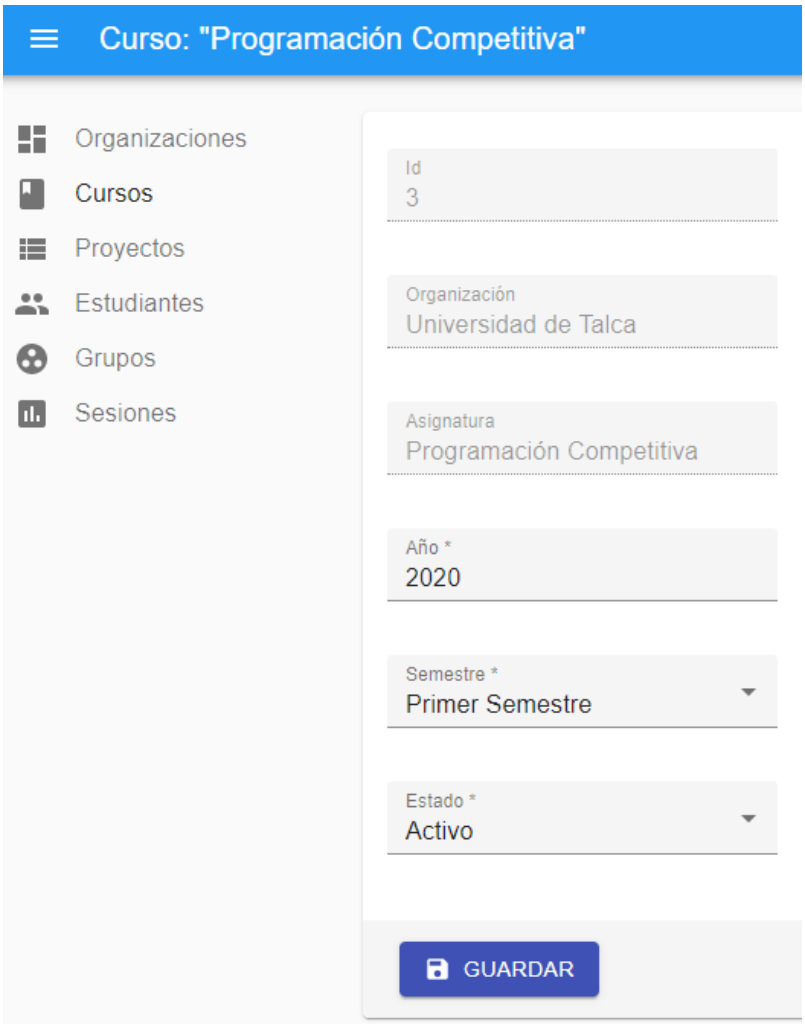


Figura A.14: Vista de Profesor para editar un Curso asociado a una Organización.

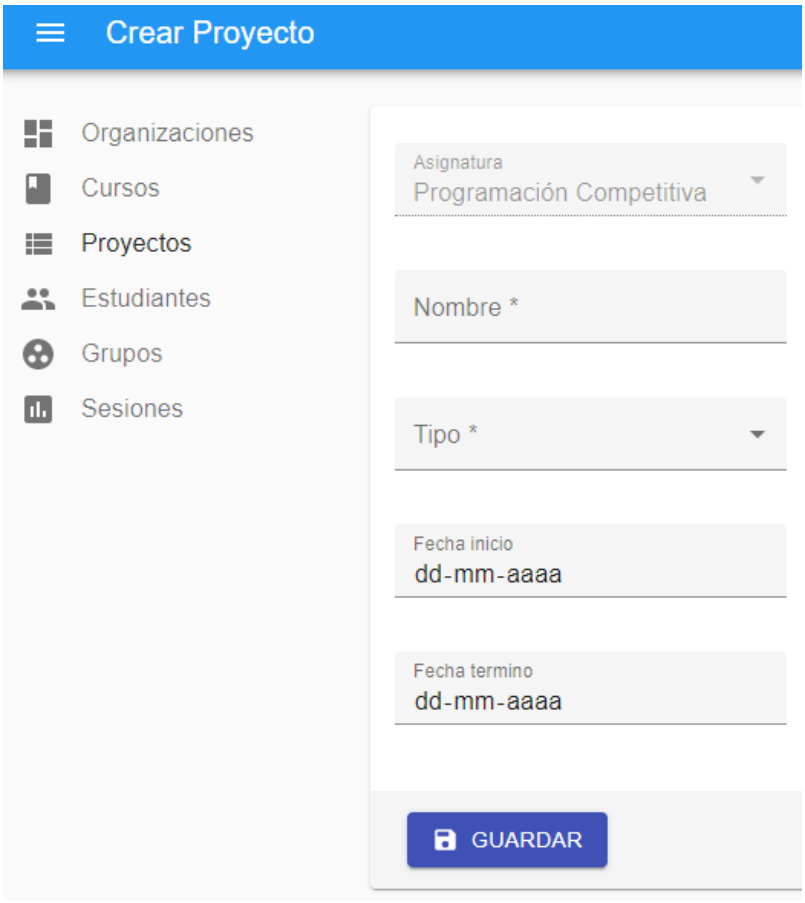


Figura A.15: Vista de Profesor para crear un Proyecto asociado a un Curso.

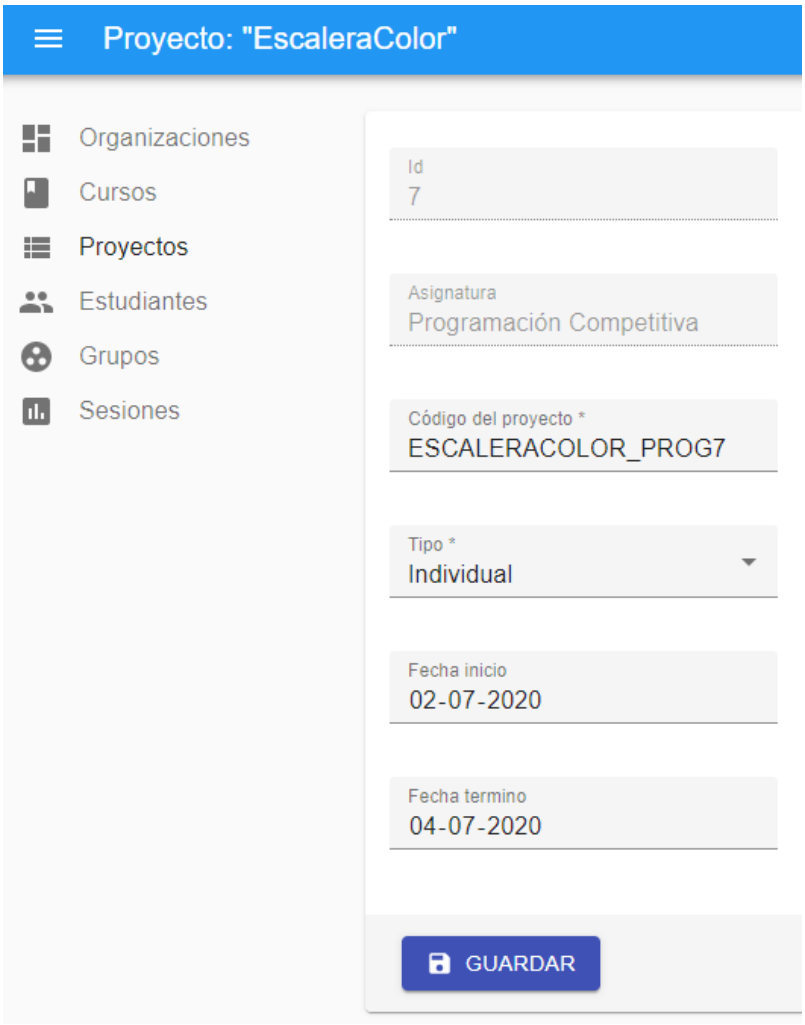


Figura A.16: Vista de Profesor para editar un Proyecto asociado a un Curso.

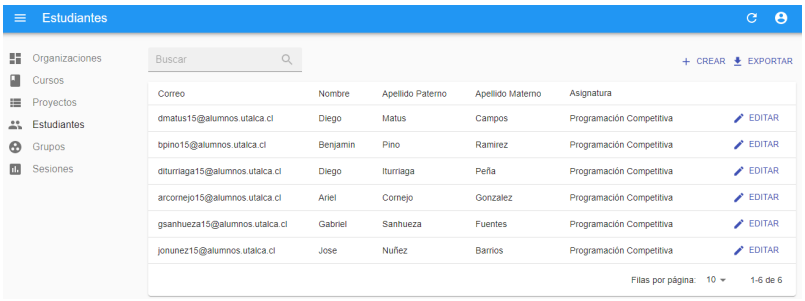


Figura A.17: Vista de Profesor para listar Estudiantes asociados a un Curso.

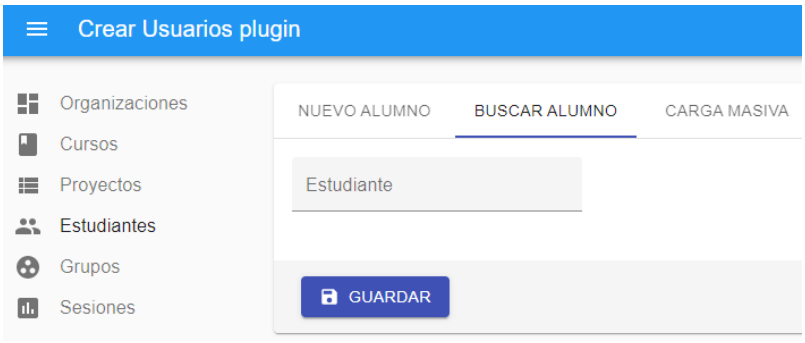


Figura A.18: Vista de Profesor para asociar un Estudiante a un Curso a través de la búsqueda por correo.

A.6. Iteración 9

Figura A.19: Vista de Profesor para agregar Estudiantes a un curso a través de un excel con extensión .xlsx.

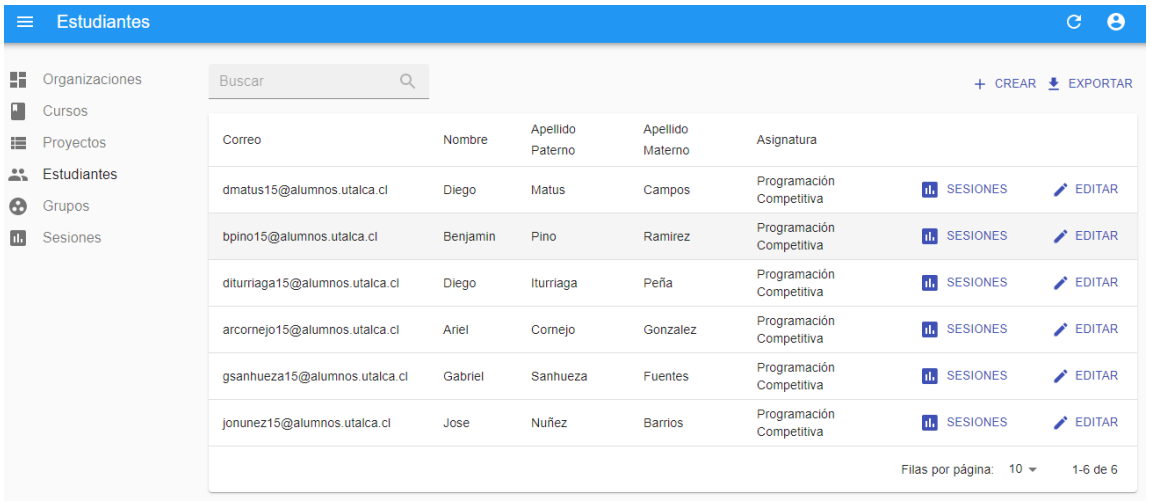


Figura A.20: Vista de Profesor para listar los Estudiantes de un Curso.

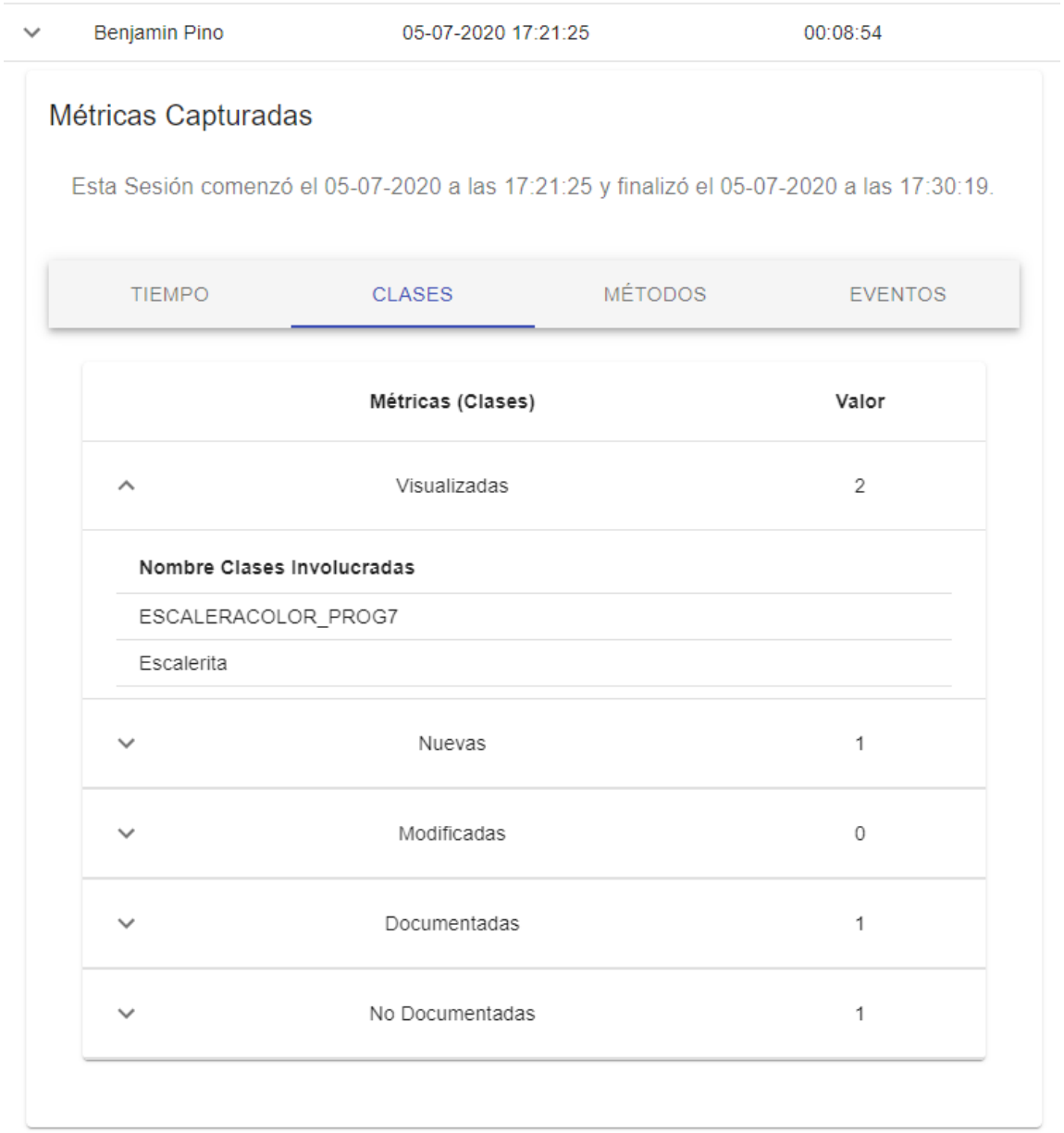


Figura A.21: Vista de Profesor para revisar las métricas de Clases.

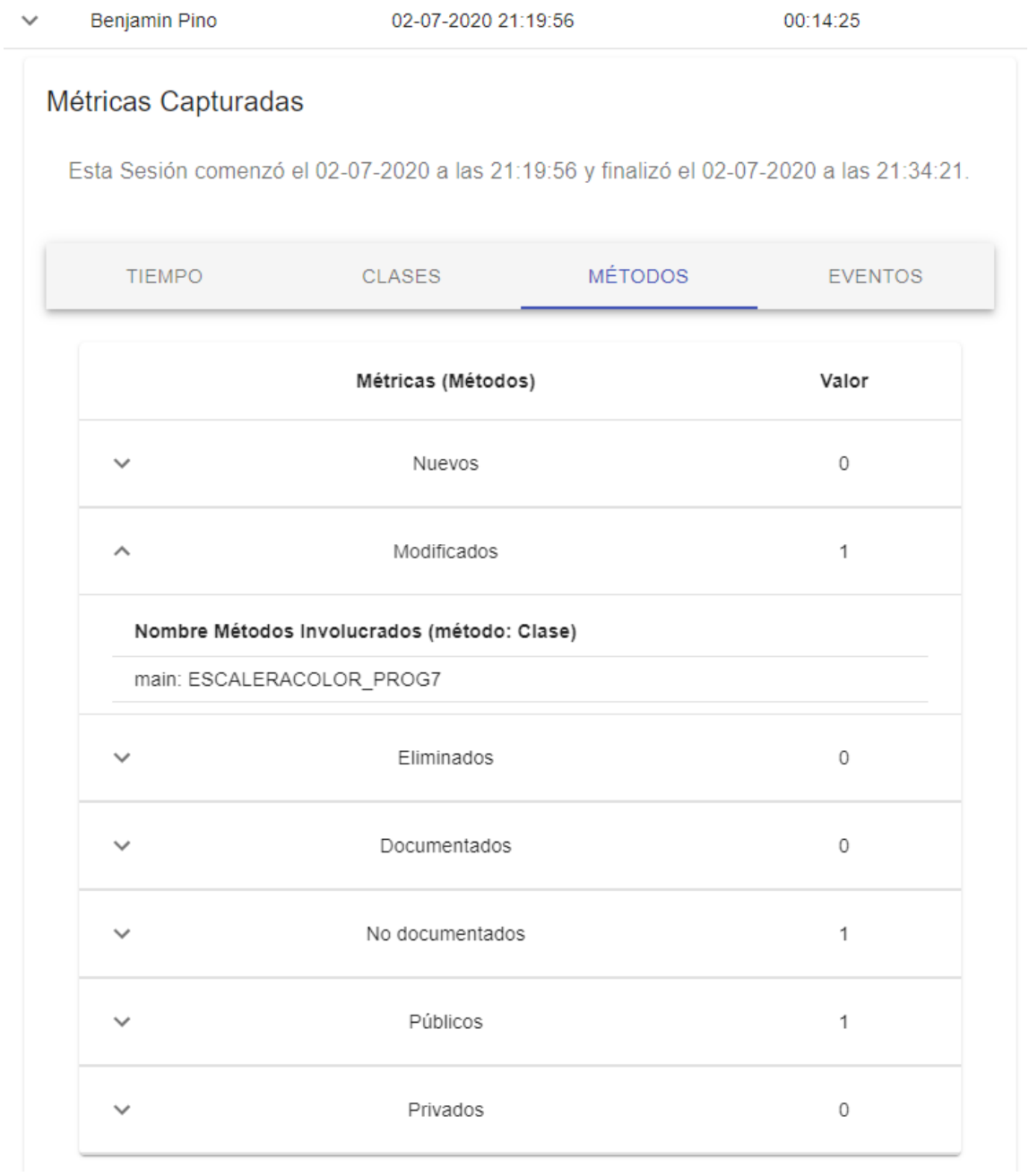


Figura A.22: Vista de Profesor para revisar las métricas de Métodos.

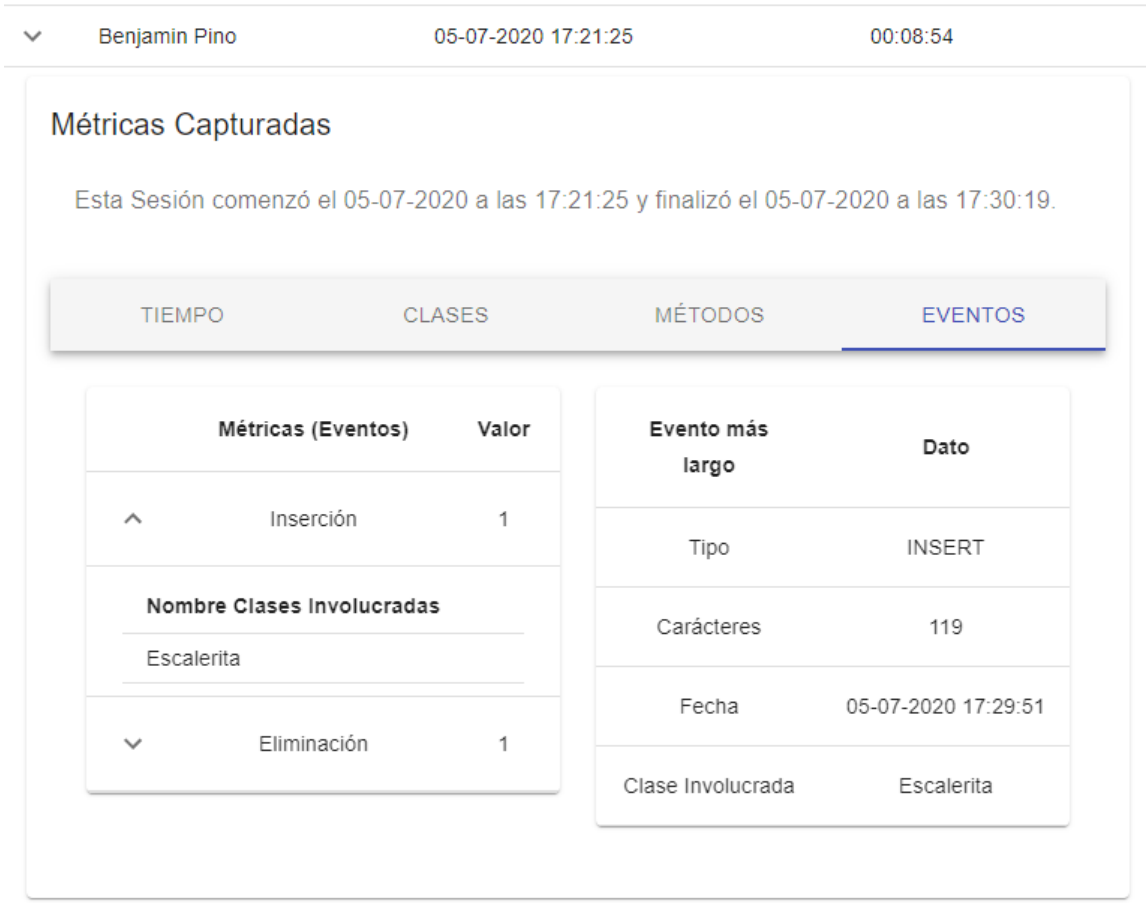


Figura A.23: Vista de Profesor para revisar las métricas de Eventos.

B. Instructivo de Instalación del Módulo

Instructivo de Instalación del Módulo para NetBeans

BlueLogger

Roberto Ureta Muñoz
rureta15@alumnos.utalca.cl
Ingeniería Civil en Computación, Universidad de Talca

1. Objetivo

El propósito de este documento es orientar al usuario en la instalación del **módulo BlueLogger** en el **IDE NetBeans**. Este tutorial fue realizado en la versión 8.2 de NetBeans.

2. Contacto

Ante cualquier duda o problema por favor, no dude en contactarme mediante mi correo institucional rureta15@alumnos.utalca.cl

3. Descarga

Importante: Para utilizar el módulo debe tener o **descargar la versión 8.2 del IDE NetBeans** disponible aquí. En la Figura 1 se visualiza el instalador sugerido a descargar.

El módulo se encuentra disponible en BlueLogger. Una vez descargado obtendrá un archivo llamado *rob-plugin.nbm*, este es el módulo que utiliza NetBeans.

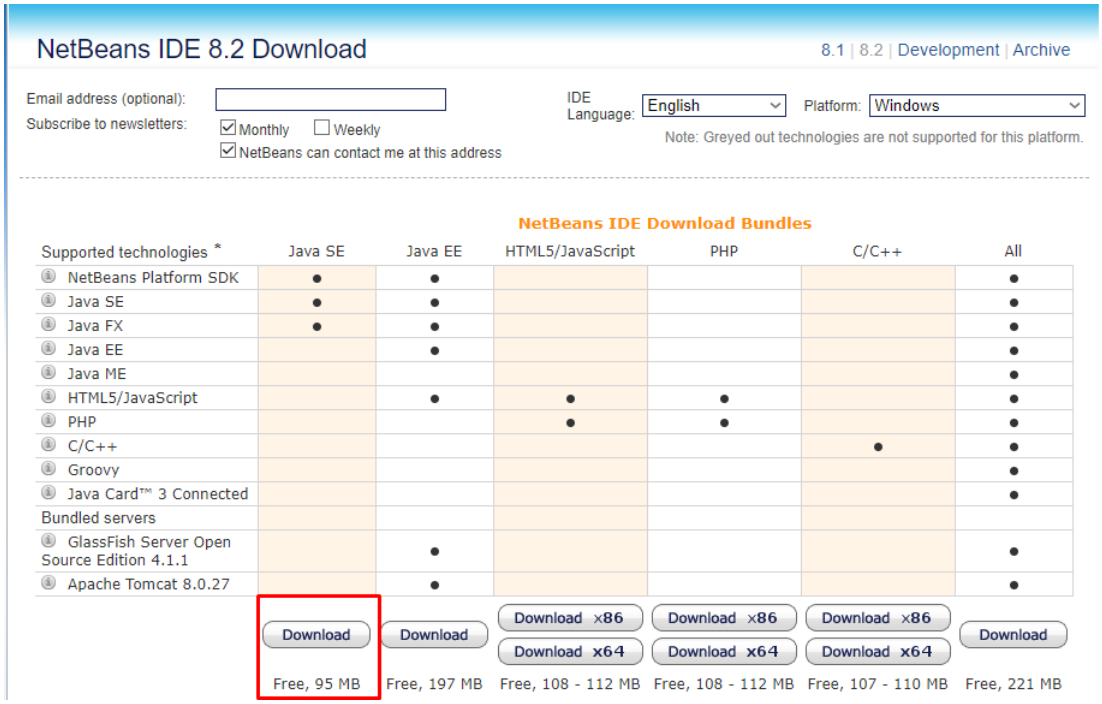


Figura 1: NetBeans 8.2.

4. Instalación

1. Abrir Netbeans y seleccionar el menú **Tools>Plugins** *(Herramientas>Módulos)*.

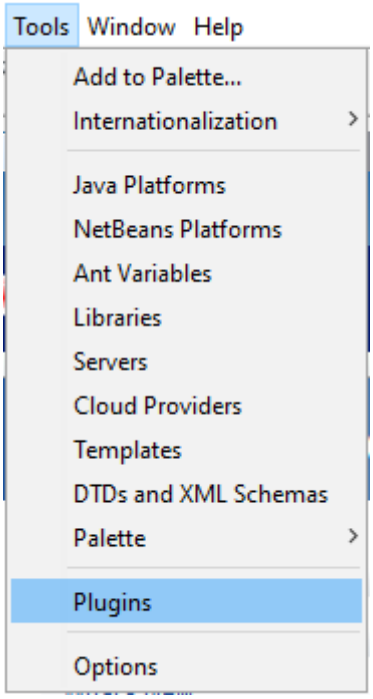


Figura 2: Menú.

2. Se abrirá una ventana. Hacer click en la pestaña **Dowloaded** *(Descargados)*.

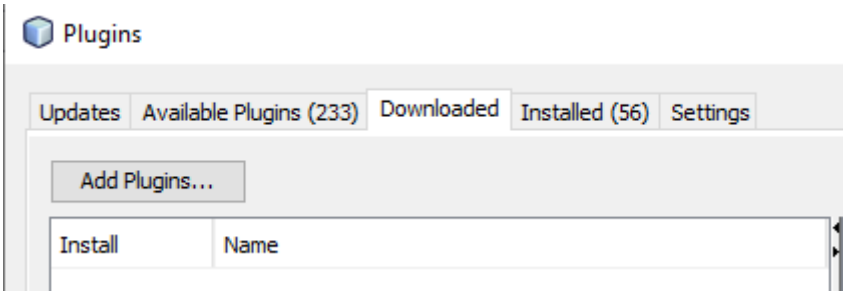


Figura 3: Pestaña Dowloaded.

3. Ahora se debe presionar sobre **Add plugins** *(Añadir módulos)* y en la ventana que se muestra, buscar el archivo descargado previamente.

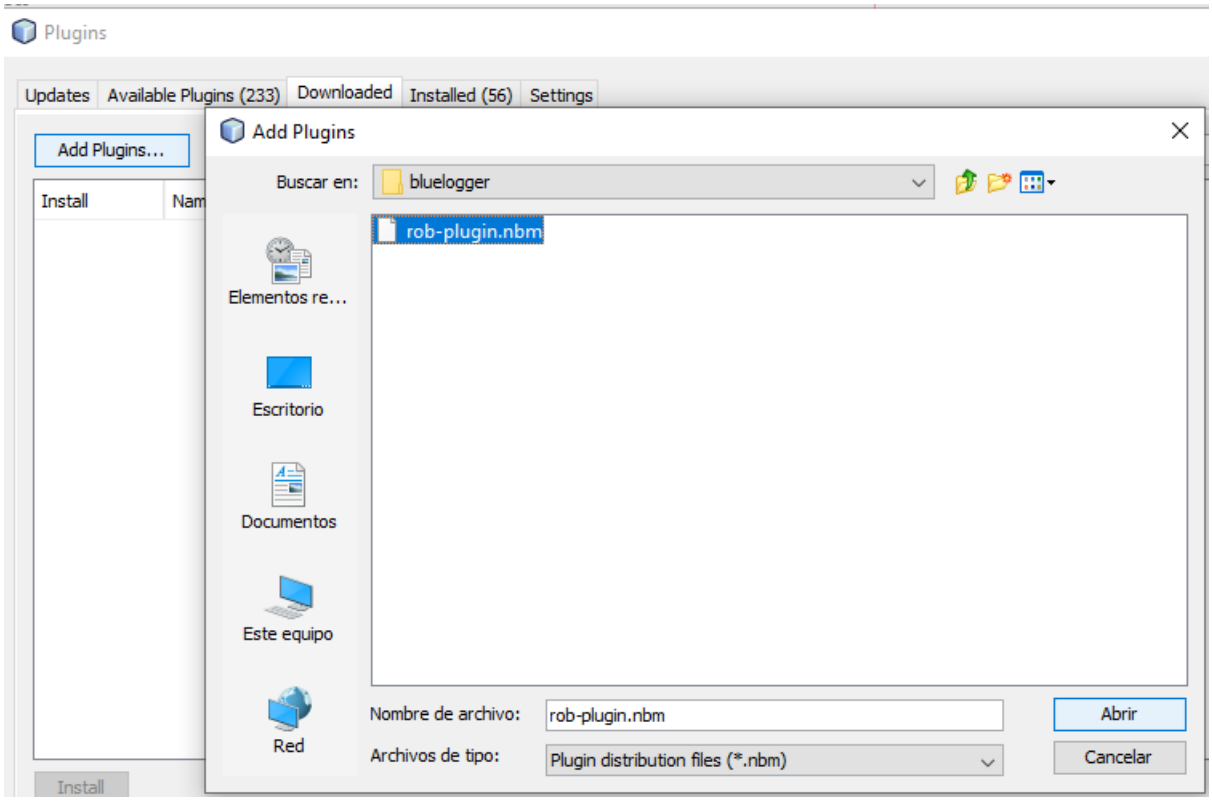


Figura 4: Selección de archivo.

4. Seleccionar BlueLogger y presionar **Install** *(Instalar)*.

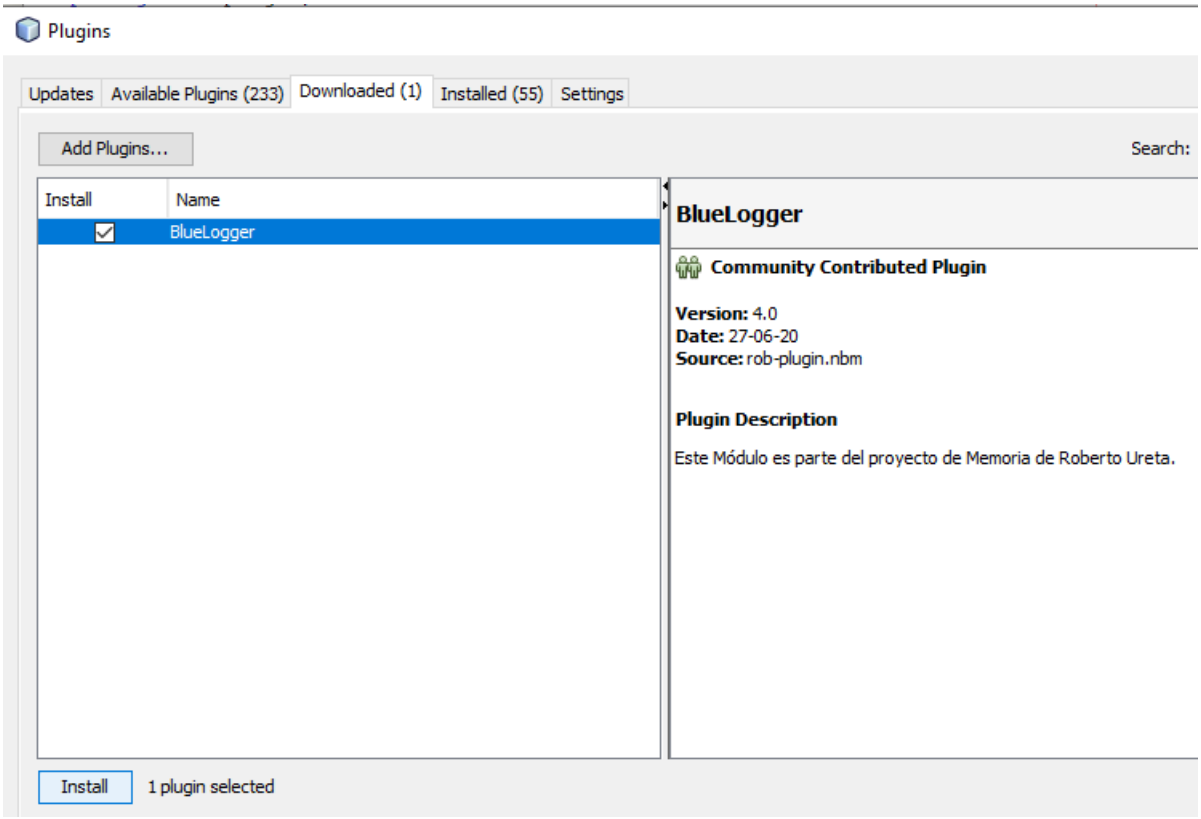
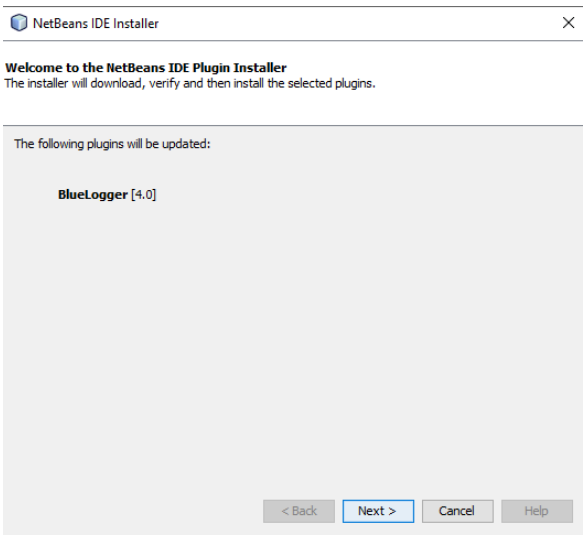
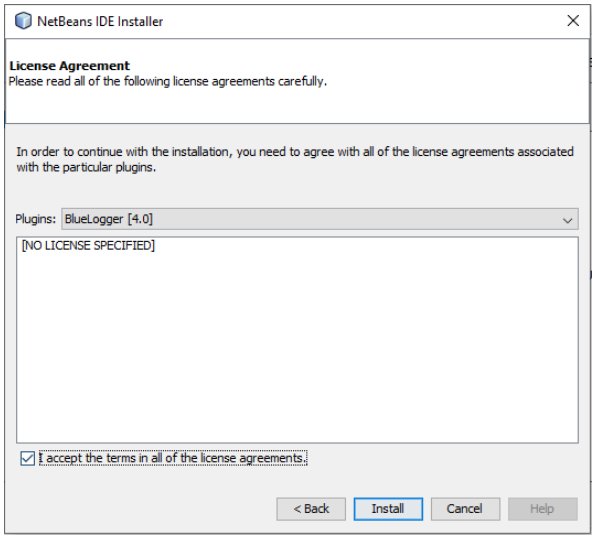


Figura 5: Presionar Install.

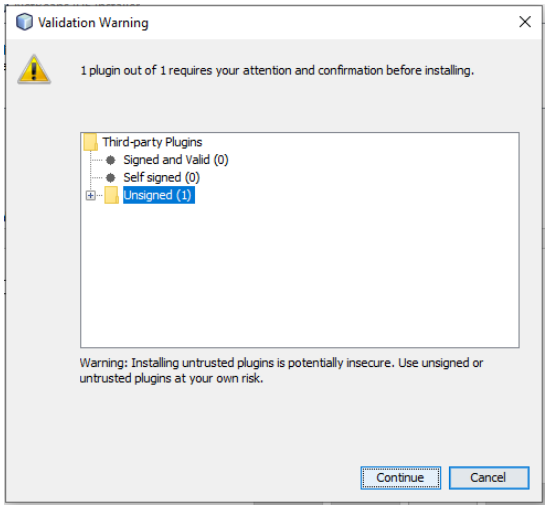
5. Ahora debe comenzar la instalación, aceptar la licencia, continuar con la advertencia de validación y por último finalizar la instalación.



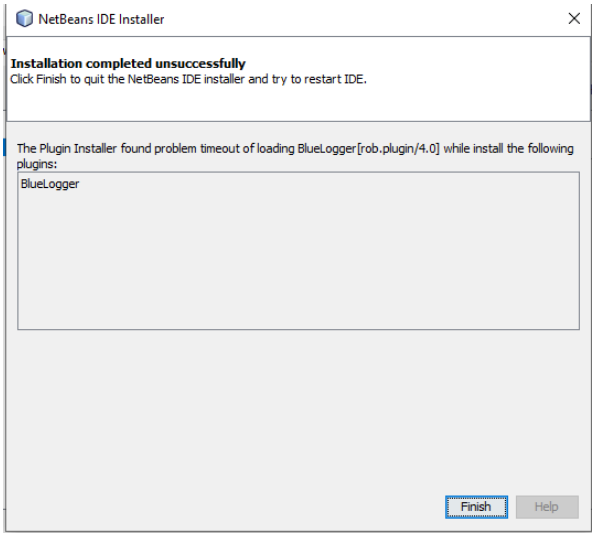
(a) Comienzo.



(b) Licencia.



(c) Advertencia.



(d) Final.

Figura 6: Proceso de Instalación.

6. Luego, en la nueva ventana debe ingresar su **correo institucional** y presionar **OK**.

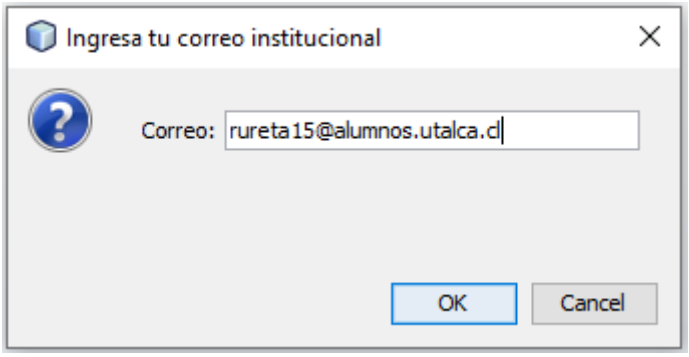


Figura 7: Correo institucional.

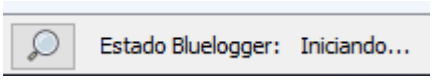


Figura 8: BlueLogger Iniciando.

7. Ingresar a su correo institucional y abrir el correo con el asunto **"Validación Usuario BlueLogger Plugin"**, copiar el código enviado para posteriormente pegarlo en la nueva ventana que aparece en NetBeans.

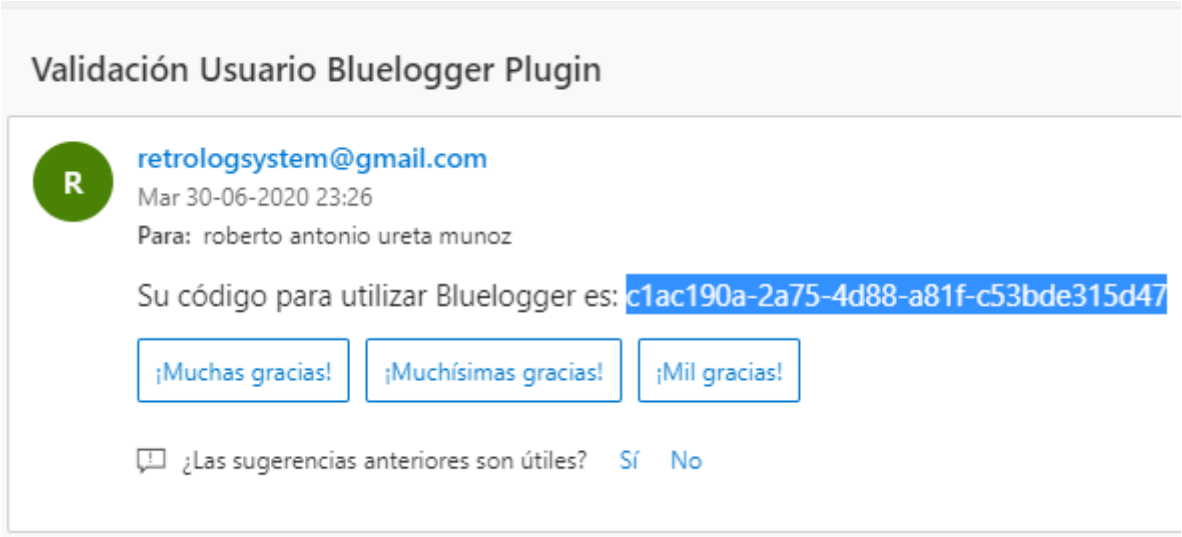


Figura 9: Correo.

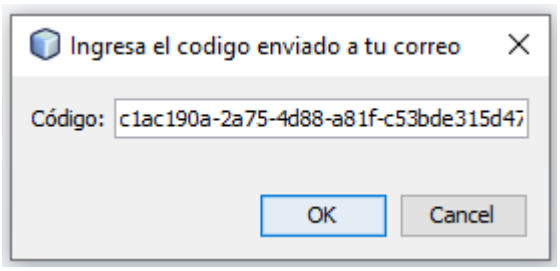


Figura 10: Código de validación.

8. Finalmente, en la esquina inferior derecha de NetBeans debiera aparecer en color verde que **BlueLogger** está **Funcionando**.

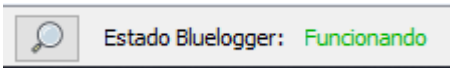


Figura 11: BlueLogger Funcionando.

5. Problemas Conocidos

Problema 1: Si aparece un mensaje como el de la Figura 12 se debe presionar el botón **Disable Modules and Continue** *(Desactivar Módulos y Continuar)*.

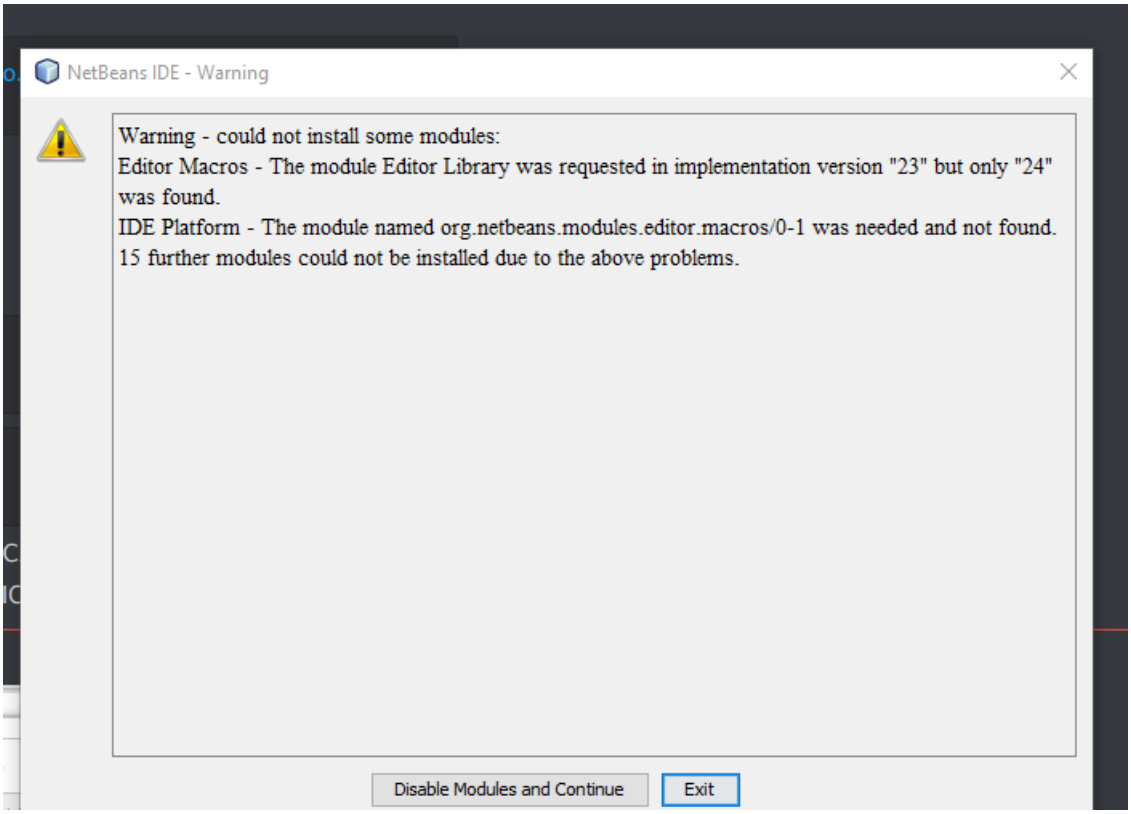


Figura 12: Problema 1.

Problema 2: Si no llega el correo de validación, una posible solución es desactivar el antivirus de su computador. En caso de que el problema persista, será necesario contactarse con el correo que aparece en la sección Contacto indicando el problema.

C. Encuesta de evaluación del módulo BlueLogger

A continuación, se incluye la encuesta utilizada para evaluar el módulo de BlueLogger, las respuestas[1] se incluyen en el pie de página.

[1] https://docs.google.com/spreadsheets/d/1V6bV_1zVT_PbzvvgMw6yUHuOAhkmz7LbLKpxFUbzwn8/edit?usp=sharing

Encuesta de evaluación del módulo BlueLogger

Se ha desarrollado un módulo disponible para el IDE NetBeans que sirve para realizar un seguimiento del trabajo realizado por los estudiantes en distintos proyectos de programación.

Esta encuesta se realiza para evaluar la funcionalidad y facilidad de uso del módulo.
*Obligatorio

Datos Generales

Ingeniería Civil en Computación - Universidad de Talca

1. Edad *

Marca una opción

Marca solo un óvalo.

- ◯ 18 - 21 años
- ◯ 22 - 25 años
- ◯ mayor a 25 años
- ◯ Otros: _____

2. Género *

Marca una opción

Marca solo un óvalo.

- ◯ Masculino
- ◯ Femenino

3. Versión de NetBeans *

Indicar la versión de NetBeans que utilizó para usar el módulo.

Marca solo un óvalo.

- 8.0
- 8.1
- 8.2
- Otros: _____

Evaluación del módulo BlueLogger

En esta sección, debe seleccionar el grado de de acuerdo o desacuerdo referente a la utilización del sistema.

4. Funcionalidad del módulo *

Marca solo un óvalo por fila.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuero ni de acuerdo	De acuerdo	Totalmente de acuerdo
NetBeans permite instalar correctamente el módulo.					
El módulo permite ingresar un correo electrónico.					
El módulo envía el código de validación al correo.					
El módulo permite ingresar el código de validación.					
El módulo permite visualizar que se encuentra funcionando.					

Evaluación del módulo BlueLogger

En esta sección, debe seleccionar el grado de de acuerdo o desacuerdo referente a la facilidad de uso del módulo.

5. Facilidad de uso del sistema *

Marca solo un óvalo por fila.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuerdo ni de acuerdo	De acuerdo	Totalmente de acuerdo
El Instructivo de Instalación es intuitivo.					
El módulo es fácil de instalar.					
La interfaz del módulo es intuitiva.					
El procedimiento para validar al estudiante es comprensible.					
La cantidad de información entregada por el módulo es suficiente.					

Preguntas

En esta sección, se solicita responder las siguientes preguntas.

6. ¿El módulo presentó algún error al momento de utilizarse? *

7. ¿Qué cambios o mejoras le haría usted al módulo? *