Universidad Autónoma de Madrid
Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

HERRAMIENTA PARA EL MODELADO PREDICTIVO EN ENTORNOS EDUCATIVOS EN LÍNEA

Autor: Víctor Macías Palla
Tutor: Ruth Cobos Pérez

Junio 2017

HERRAMIENTA PARA EL MODELADO PREDICTIVO EN ENTORNOS EDUCATIVOS EN LÍNEA

Autor: Víctor Macías Palla
Tutor: Ruth Cobos Pérez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2017

Resumen

Resumen

El uso de las nuevas Técnologías de la Información y de la Comunicación (TIC) se han convertido en una herramienta muy versátil en todo tipo de campos. Fomentar su uso en el ámbito del aprendizaje y la educación es de vital importancia.

Los cursos MOOC (Massive Open Online Course) es uno de los ejemplos más importantes de integración de las TIC en el campo del aprendizaje y la educación.

El objetivo de este Trabajo de Fin de Grado consiste en la creación de una aplicación modular y escalable que, a partir de los datos recogidos de MOOCs, permita aplicar de forma sencilla un proceso completo de Data Science, donde se aplicarán algoritmos de Machine Learning para generar modelos predictivos con los que poder predecir qué estudiantes han aprobado el curso y obtenido por tanto un certificado.

La aplicación desarrollada ayuda al usuario a analizar y comparar la calidad de predicción de distintos algoritmos de Machine Learning, permitiendo la posibilidad de exportar los resultados para realizar a posteriori un análisis más exhaustivo.

Palabras Clave

Analíticas de Aprendizaje, Ciencia de datos, Cursos Online Masivos y Abiertos, indicadores, analizar y predecir

Abstract

The use of Information and Communication Technologies (ICT) has become a versatile tool in different types of fields. Promoting their use in the field of learning and education is of vital importance.

The Massive Open Online Courses (MOOCs) is one of the most important examples of ICT integration in the field of learning and education.

The objective of this End-of-Grade Study is the creation of a modular and scalable application that, based on the data collected from a MOOC course, allows the simple application of a complete Data Science process, where Machine Learning algorithms will be applied to generate predictive models which could be used to predict which students have passed the course and therefore obtained a certificate.

The developed application helps the user to analyze and compare the prediction quality of different Machine Learning algorithms, allowin the possibility of exporting the results in order to do a more exhaustive analysis.

Key words

Learning Analytics, Data Science, Massive Open Online Courses, indicators, analyze and predict

Agradecimientos

Me gustaría agradecer a Ruth Cobos por darme la oportunidad de desarrollar este proyecto y por todo el apoyo y ayuda recibida.

Gracias a mis padres por preocuparse por mí todo este tiempo que me han apoyado desde la distancia durante toda la carrera, ha sido difícil vivir alejado de ellos en la capital. Gracias al resto de mi familia, a mis abuelos Luciano, Almudena, Antonio y Alicia, a mis primos y a mis tíos.

Agradecer a mis compañeros de piso, Sergio, Jesús y Sara, que he ido teniendo a lo largo de la carrera y con los que se han pasado grandes momentos.

Por supuesto agradecer a todos los compañeros de la carrera, Harry, Antonio, Carlos, Gonzalo, Lara, Alfonso... y a todos los que no he nombrado, pero que sabéis que estáis en la lista.

Por último saludar a mis amigos de la infancia que siempre han estado ahí: Maci, Iván, Belén.

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

En este capítulo se expone la motivación que ha llevado elegir este tema como Trabajo de Fin de Grado, así como la introducción a la temática del mismo. Además, se muestra un análisis de objetivos a cumplir en forma de hitos para la realización del proyecto. Por último, se detalla la estructura que tiene este documento.

1.1. Motivación del proyecto

Los MOOCs (Massive Open Online Courses) [1] han emergido en los últimos años como cursos en línea gratuitos en los que cualquier persona puede inscribirse. Por esta razón, este tipo de cursos suelen tener un gran número de inscritos, que pueden llegar hasta los cientos de miles. La aparición de estos cursos nos lleva a estudiar un nuevo paradigma de educación a distancia, en el cual el desarrollo tecnológico está cobrando una gran importancia, ya que debido a la gran cantidad de inscritos, surgen problemas de escalabilidad, y aparecen nuevos retos a la hora de diseñar nuevos contenidos educativos.

Por otra parte, las analíticas de aprendizaje (Learning Analytics) [2] también ha cobrado gran importancia en los últimos años debido a la necesidad de analizar y extraer información relevante a partir de todos los datos que se recogen en las plataformas de servicios educativos. En particular, las analíticas de aprendizaje engloban la recogida de datos recogidos a partir de las interacciones de los alumnos y del propio curso, su análisis y comprehensión, con el objetivo de mejorar el proceso educativo.

Este contexto ha llevado a la Universidad Autónoma de Madrid [3] a incorporarse en 2014 consorcio edX [4] para difundir a través de esta plataforma sus cursos MOOCs, y en 2015 se publicaron los primeros 4 MOOCs de la UAM. Esta plataforma guarda todos los datos recogidos a partir de la interacción de cada estudiante en el curso en el que está apuntado, y los facilita a las distintas entidades académicas para aplicar sobre ellos tácticas de learning analytics.

Desde 2015 se han ido desarrollando distintos proyectos relacionados con los cursos MOOC de la UAM, que tienen que ver con:

- Estudio de la estructura de ficheros de eventos generados.
- Generación de indicadores.
- Desarrollo de cuadros de mando para ver estadísticas concretas de los cursos.
- Estudio del estado del arte de los MOOCs y las Learning Analytics.
- Uso de Machine Learning para predecir usuarios aprobados con certificado.

Además, se han presentado distintos artículos en conferencias importantes de Learning Analytics a nivel mundial, como en el congreso LAK (Learning Analytics and Knowledge) [5] [6], la conferencia EDUCON (Global Engineering Education Conference) [7], la cumbre eMOOCs (European MOOC Stakeholder Summit) [8] [9] [10], la conferencia SIIE dentro del CEDI 2016 [11] o la conferencia L@S (ACM Conference on Learning @ Scale) [12]

Por este motivo, y gracias a toda la experiencia adquirida en estos trabajos previos, ha surgido la necesidad de crear una aplicación que integre todo el proceso de learning analytics que se aplicará sobre los datos obtenidos de los cursos MOOC que tiene la UAM en la plataforma edX, de forma que se puedan extrapolar a cualquier curso MOOC alojado en esa plataforma.

1.2. Objetivos

El objetivo de este proyecto es el desarrollo de una aplicación a la que denominaremos edX-MAS (edX Model Analizer System), desde la toma de requisitos hasta el desarrollo, pruebas e integración. Esta aplicación integrará todo el proceso de learning analytics aplicado a los cursos MOOC de la plataforma edX, y usará estas técnicas para guardar y comparar distintos modelos predictivos que se utilizarán para predecir absentismo o abandono del curso y obtención de certificados que certifican que se ha superado el curso.

Para poder completar este proyecto, se deben cumplir los siguientes objetivos y metas:

1. Analizar los paquetes de datos de los cursos MOOC facilitados desde la plataforma edX, los cuales están anonimizados.
2. Realizar un estudio sobre el aprendizaje electrónico, los cursos online masivos y abiertos y las analíticas de aprendizaje.
3. Realizar un estudio sobre las tecnologías utilizadas en la actualidad en estos campos, incluyendo tipos de bases de datos, lenguajes de programación, librerías y técnicas de Machine Learning.
4. Investigar la existencia de otras aplicaciones, herramientas o proyectos similares.
5. Realizar un análisis de requisitos con los requisitos funcionales y no funcionales que debe cumplir la aplicación.

6. Diseñar la aplicación. El diseño debe incluir:
   a) El modelo de base de datos utilizado y sus características.
   b) Diagramas de clases de las distintas partes en las que se dividirá la aplicación.
7. Desarrollar la aplicación, para lo cual la dividiremos en distintos hitos:
   a) Almacenamiento de datos básicos relacionados con el curso y los usuarios, y generación y almacenamiento de distintos indicadores a partir de los paquetes de datos. A este proceso lo denominaremos "importación del curso".
   b) Realización de predicciones sobre obtención de certificados utilizando los datos almacenados previamente, utilizando modelos a partir de algoritmos de Machine Learning. A este proceso lo denominaremos "creación del modelo".
   c) Generación y almacenamiento tanto de los modelos como de las estadísticas asociadas al mismo. A este proceso lo denominaremos "generación de estadísticas".
   d) Exportación y visualización de las estadísticas generadas de un modelo predictivo. A este proceso lo denominaremos "exportación y visualización"
   e) Integración de los distintos procesos descritos anteriormente en una aplicación web.
8. Realizar pruebas de integración y evaluación con datos reales, comparando los resultados con otros estudios realizados.
9. Realizar un estudio con las posibles mejoras de la aplicación, así como los posibles trabajos futuros que se podrán derivar a partir de este proyecto.
10. Escribir la memoria del Trabajo de Fin de Grado.

1.3. Estructura del documento

Este documento está organizado en distintos capítulos en los que se detallan todos los aspectos del proyecto.

En este *primer capítulo* se ha realizado una breve introducción al proyecto desarrollado, exponiendo las motivaciones y los objetivos principales que se quieren alcanzar.

En el *segundo capítulo* se presenta el Estado del arte, donde se realiza un análisis del contexto en el que se introducen los conceptos de aprendizaje electrónico, curso en línea masivo y abierto, y analíticas de aprendizaje. Este estudio incluye también un estudio sobre las tecnologías que se pueden usar, y las que se van a utilizar.

En el *tercer capítulo* se expone la funcionalidad y el Análisis de Requisitos de la aplicación, así como su división en módulos.

En el *cuarto capítulo* se presenta el diseño de la aplicación, incluyendo distintos diagramas que ayuden al desarrollo.

En el *quinto capítulo* se corresponde con el desarrollo e implementación de la aplicación.

En el *sexto capítulo* se explican las pruebas realizadas y los resultados obtenidos.

Finalmente, en el *octavo capítulo* se presentan las conclusiones finales del proyecto y las propuestas para futuros trabajos.

Adicionalmente, se incluyen los siguientes anexos:

Anexo A - Estructura de paquetes de datos de edX, donde se detallan las características de todos los datos que nos ha proporcionado la plataforma edX, y el tratamiento realizado para el uso de los mismos.

Anexo B - Manual de usuario de la aplicación, en el que se explica brevemente los pasos a seguir para realizar las distintas funciones que se pueden realizar con la aplicación, mostrando capturas de pantalla.

Anexo C - Encuesta de usabilidad y de evaluación del servicio, en el que se muestra el cuestionario utilizado para medir la usabilidad de la aplicación.

2
Estado del arte

En este capítulo veremos tanto el contexto como las tecnologías aplicadas al campo del aprendizaje electrónico y los MOOC, así como diferentes estudios realizados previamente.

2.1. Análisis del contexto

En esta sección realizaremos una introducción al aprendizaje electrónico y nos centraremos en los MOOC y las analíticas de aprendizaje. En particular, los definiremos y estudiaremos qué problemas han surgido en este entorno y cómo se están tratando de resolver.

2.1.1. Aprendizaje electrónico (e-learning)

El aprendizaje electrónico, más conocido por su término inglés e-learning, es una modalidad de educación a distancia que se lleva a cabo en entornos virtuales (sobre todo en Internet) y que se sirve de la tecnología para llevar a cabo todo el proceso de enseñanza - aprendizaje.

Algunas de las ventajas que presenta el aprendizaje electrónico con respecto al aprendizaje tradicional son:

- Permite realizar cualquier tipo de formación académica sin importar la ubicación.
- Ofrece al estudiante flexibilidad horaria, con todas las ventajas que conlleva (compatibilizar el aprendizaje con la vida familiar o con el trabajo, obtener un ritmo de aprendizaje personalizado, etc)
- Los cursos permiten conocer a otros estudiantes con los mismos intereses.
- Permite la exploración e investigación de nuevas formas y procesos de aprendizaje.

Un estudio más detallado sobre el estado del arte del aprendizaje electrónico se puede encontrar en [13]

2.1.2. Curso en línea masivo y abierto (MOOC)

Los Cursos en Línea Masivos y Abiertos, más conocidos por su término inglés Massive Online Open Courses (MOOC's) [1] [14], son una modalidad de educación abierta a distancia dirigida a un amplio número de participantes a través de internet.

Para que un curso online sea considerado MOOC, éste debe cumplir ciertos requisitos:

- Debe tener una estructura organizada de un curso: tener el contenido dividido en distintas temáticas, contener actividades y problemas para poner a prueba lo aprendido, y evaluaciones o pruebas que acrediten el aprendizaje adquirido.
- Debe tener carácter masivo: en principio, el número de posibles matriculados debe ser, salvo por limitaciones tecnológicas, ilimitado.
- Debe ser en línea: todo el contenido está en Internet, siendo este el principal medio de comunicación y acceso.
- Debe ser abierto: todo el contenido debe poder ser accesible de forma gratuita para todo el mundo a través de Internet.

En la actualidad, existen distintas plataformas de MOOCs que ofrecen la posibilidad de subir estos cursos a Internet, ademas de guardar un registro completo de las acciones tomadas por los estudiantes. Las más importantes son Coursera [15], edX [4] y Future-Learn [16]. La Universidad Autónoma de Madrid cuenta en la actualidad con 12 cursos MOOC que hospeda en la plataforma edX.

Desde el primer curso MOOC lanzado en 2011 por la Universidad Stanford [17], han surgido distintos problemas, los cuales han derivado en diferentes investigaciones. Por una parte, uno de los problemas principales que presentan los MOOCs son problemas relacionados con la escalabilidad, debido precisamente a la gran cantidad de estudiantes que puede haber en un curso, sobre todo al guardar un registro de las acciones de cada estudiante del curso. Por otra parte, el problema principal que presentan los MOOC es la altísima tasa de abandono, que puede sobrepasar en algunos casos más del 90 %. Esto ha motivado a realizar distintas líneas de investigación para averiguar por qué ocurre este hecho, y se contemplan posibilidades como la forma en que se implementan los cursos, la experiencia de usuario que provocan, condiciones culturales y tecnológicas de los alumnos, etc.

Un estudio mucho más detallado del estado de arte de los MOOCs se puede encontrar en [11].

2.1.3. Analíticas de aprendizaje (Learning Analytics)

Debido al éxito que ha tenido el aprendizaje electrónico, y en particular al éxito que han tenido los MOOCs, surge la necesidad de registrar y, posteriormente, estudiar los datos que tienen que ver con la actividad de los estudiantes.

Siempre que se dispone de un conjunto de datos de los cuales se quiere extraer información, aparece la Ciencia de Datos. La Ciencia de Datos (Data Science) es un campo interdisciplinario que involucra diferentes métodos para extraer información sobre datos estructurados o no estructurados [18].

No existe una definición exacta sobre la secuencia de pasos a seguir en un problema de Data Science, aunque se puede tomar como punto de partida los pasos mostrados en la figura 2.1, que está basada en el siguiente documento [19].

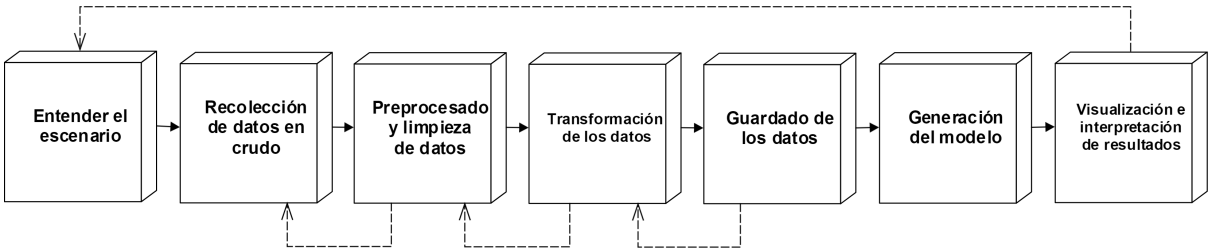

Figura 2.1: Secuencia básica en un problema de Data Science

Las analíticas de aprendizaje, más conocidas por su término inglés Learning Analytics, consisten en la aplicación de técnicas de Ciencia de Datos sobre los datos provenientes de los cursos online. Vamos a describir brevemente los pasos a seguir en un problema de Learning Analytics, tal y como se muestra en la figura 2.1:

1. **Entender el escenario**: El primer paso en todo problema de Data Science es estudiar y entender el escenario del cual se ha de realizar el estudio. En el caso de Learning Analytics, el escenario va a ser todo el entorno relacionado con los cursos y sus características. Además, se han de estudiar los distintos tipos de datos que se disponen, su estructura interna y su formato, así como qué resultados queremos obtener.

2. **Recolección de datos en crudo**: El siguiente paso consiste en obtener todos los datos que se van a utilizar para el estudio. En un problema de Learning Analytics, los datos a utilizar serán todos los datos relacionados con el curso y los relacionados con los estudiantes del curso: resultados finales, listado de eventos generados, información de foros, etc.

3. **Preprocesado y limpieza de datos**: El siguiente paso consiste en preprocesar los datos. Este proceso engloba clasificar y agrupar los distintos tipos de datos, eliminación de datos poco relevantes y organizar los datos de manera apropiada para facilitar su guardado y acceso.

4. **Transformación de los datos**: Una vez preprocesados los datos, es necesario transformarlos para poder generar un modelo. Al final de la transformación se han tenido que generar lo que denominaremos Variables de entrada o Indicadores. Un indicador es un par clave - valor que se utilizará como entrada en el modelo. Este par clave - valor depende del problema que se quiera estudiar. En Learning Analytics típicamente se quieren encontrar patrones entre estudiantes (en este caso la clave será el identificador del estudiante) o características del propio curso (en este caso

la clave será el identificador del curso) del que analizar características. Es muy importante generar un conjunto de indicadores bien definido para elaborar un buen modelo. Un buen indicador ha de ser sencillo de generar y con un significado que se pueda interpretar.

5. **Guardado de los datos**: Tras preprocesar y transformar los datos, éstos se deben persistir. Para ello se debe diseñar previamente cómo se van a almacenar, diseñando los diagramas correspondientes (diagrama de Entidad-Relación en caso de guardarlos en una base de datos relacional, por ejemplo). En todo el proceso de preprocesado,limpieza,transformación y guardado, es muy importante estudiar qué datos queremos persistir y qué datos no, analizando las ventajas e inconvenientes de esta decisión.

6. **Generación del modelo**: Una vez se han almacenado y persisitido los datos, se han de generar los modelos predictivos. Todo modelo predictivo consta de un conjunto de variables de entrada y una salida, y su objetivo es aprender, a partir de los valores de las variables de entrada, su correspondiente salida. Además, todo modelo predictivo tiene asociado un algoritmo de aprendizaje, por lo que podremos clasificar los distintos modelos predictivos con los distintos enfoques de aprendizaje que se utilizan en el campo del Aprendizaje Automático (en inglés, Machine Learning). En general, todo modelo predictivo consta de dos fases: el entrenamiento y el test, y para cada una de las cuales se utiliza una partición del conjunto de datos de entrada.

7. **Visualización e interpretación de resultados**: Una vez generado el modelo, es necesario conocer lo bueno que es comparándolo con otros modelos existentes. Para ello usaremos una combinación de:

   - Calidad del modelo. Lo más importante de un modelo es minimizar al máximo el error de predicción, por lo que un modelo será mejor que otro si tiene menos errores.
   - Tiempo que tarda el modelo en realizar el entrenamiento/test. En caso de que dos modelos tengan una tasa de acierto muy similar, se compararán los dos modelos con esta medida.

   En Aprendizaje Automático en general, la forma de medir la calidad de un algoritmo de aprendizaje depende del tipo de variable de salida que se quiere predecir. En este proyecto en particular, todas las variables de salida son de tipo booleano, por lo que nos centraremos en explicar solamente este tipo de variables.

   En general, existen distintas medidas para medir la calidad para este tipo de modelos, como el F1-score [20] o la accuracy [21]. Sin embargo vamos a utilizar como medida de calidad la medida más usada para variables de salida booleanas, que es el AUROC [22] (Area Under the Receiver Operating Characteristics curve), AUC para simplificar. Para calcularla se parte de lo que se denomina espacio ROC [23], un cuadrado de dimensiones $1x1$ cuyo vértice inferior izquierdo se sitúa en el punto $(0, 0)$ del plano en el que el eje X es la tasa de fracasos y el eje Y es la tasa de éxitos. En el espacio ROC se situarán las curvas ROC, que se generarán a partir del modelo, y miden la calidad de un modelo en función de lo cerca que están los

puntos de la curva de la recta $y = 1$. A partir de una curva ROC se calcula el AUC, que es el área que deja la curva ROC por debajo.

Además de comparar qué modelo es mejor, es interesante conocer la importancia que han tenido las variables de entrada en el propio modelo. Calcular esta medida depende del tipo de modelo predictivo, por lo que hay que tener en cuenta qué modelo predictivo se está utilizando para poder obtener esta medida.

El uso de analíticas de aprendizaje puede ser de gran valor, tanto para profesores como para las instituciones en diversos campos de acción. Por un lado, permiten a los profesores una fuente de retroalimentación que les permite adaptar los contenidos del curso y obtener un contenido de mayor calidad. Por otro lado, realizar predicciones sobre estudiantes en riesgo de abandono podría dar alguna idea sobre este problema que presentan estos cursos.

2.2. Estudio de tecnologías

En esta sección realizaremos una breve comparativa entre las alternativas tecnológicas que se han tenido en cuenta a la hora de desarrollar el proyecto, argumentando la elección de las que van a ser utilizadas en la aplicación. Se realizará dicho estudio para las bases de datos, lenguajes de programación, librerías y modelos predictivos.

2.2.1. Modelos de bases de datos

Los sistemas de bases de datos pueden ser relacionales o no relacionales.

Bases de datos relacionales

Las bases de datos relacionales [24] son aquellas que adoptan el modelo relacional. Están formadas por un conjunto de tablas cuyos campos están predefinidos, lo que permite realizar relaciones entre tablas. A estas bases de datos se le asocia el lenguaje de consultas SQL [25]

Los sistemas de gestión de bases de datos relaciones más comunes son: MySQL [26], Oracle 12g [27], SQLite [28] y PostgreSQL [29]

Bases de datos no relacionales

Las bases de datos no relacionales, más conocidas como NoSQL [30], engloban al resto de bases de datos que difieren en algunos de los aspectos básicos de las bases de datos relacionales, como no tener esquemas predefinidos, carecer de relaciones, no usar el lenguaje de consultas SQL, etc. Estas bases de datos surgieron debido a los problemas de velocidad que presentaban las bases de datos relacionales a la hora de tratar grandes cantidades de datos.

Los sistemas de gestión de bases de datos no relacionales más comunes en la actualidad son: MongoDB [31], Cassandra [32] y CouchDB [33].

Comparativa entre modelos de bases de datos

A continuación realizaremos una comparativa entre las bases de datos relacionales y no relacionales, mostrando sus ventajas e inconvenientes.

- Ventajas de las Bases de Datos relacionales
  - Su uso está más adaptado. Debido al largo tiempo que llevan en el mercado tienen un mayor soporte.
  - Las operaciones realizadas sobre la base de datos son atómicas.
  - Garantizan la integridad de los datos y evitan duplicidad.
- Desventajas de las Bases de Datos relacionales
  - La atomicidad de las operaciones puede derivar en problemas de rendimiento.
  - Los problemas de escalabilidad son su punto débil.
- Ventajas de las bases de datos no relacionales
  - Soporta estructuras distribuidas.
  - Mayor adaptabilidad a cambios relacionados con el proyecto.
  - Escalabilidad horizontal, permitiendo crecer aumentando el número de máquinas.
  - Optimizado para grandes cantidades de datos.
- Desventajas de las bases de datos no relacionales.
  - En general no soportan la atomicidad de las operaciones.
  - No son compatibles al 100 % con SQL.
  - No garantizan la unicidad de los datos.
  - Al llevar menos tiempo en el mercado, el soporte es de peor calidad.
  - Falta de estandarización.

En la aplicación se ha optado por utilizar el sistema de gestión de bases de datos PostgreSQL. Se ha elegido este sistema de gestión de entre los posibles sistemas de gestión de bases de datos relacionales ya que se ha trabajado previamente con él, es Open Source y sencillo de utilizar.

Además, se ha elegido una base de datos relacional frente a una no relacional debido a que la aplicación no aprovecharía las ventajas que presentan las bases de datos no relacionales. Esto se debe a que se ha decidido almacenar en la base de datos los datos una vez realizados los procesos de preprocesado, limpieza y transformación, permitiendo así un diseño relacional y ahorrando grandes cantidades de memoria en disco. El ejemplo más relevante es el almacenamiento de la información relativa a los eventos generados a partir de la interacción del usuario, de los cuales solamente se guardarán los indicadores. El resto de datos (curso, edición usuarios, indicadores, certificados), que no necesitan un proceso de transformación, también son muy apropiados para aplicarles un modelo entidad-relación, por lo que es conveniente guardarlos en una base de datos relacional.

Para el almacenamiento de las estadísticas de los modelos hubiera sido más apropiado una base de datos no relacional, ya que esta información es muy fácil estructurarla en formato JSON. A pesar de ello, decidimos optar por guardarla en la base de datos relacional ya que tener dos bases de datos separadas y con características tan distintas podría poner en riesgo la integridad de los datos y tener impacto negativo en la aplicación final.

Por último, es posible que surjan problemas de rendimiento en las queries conforme se guarden más cursos e indicadores. Para solventar estos problemas de rendimiento, PostgreSQL cuenta con los índices, que optimizan ciertas partes de una consulta a cambio de utilizar más recursos en disco y, por tanto, añadir complejidad a la hora de insertar. Si los problemas de rendimiento son graves, bastaría con crear un índice en la tabla de indicadores para cada curso.

2.2.2. Lenguajes de desarrollo y librerías

Tras una investigación sobre los lenguajes de desarrollo típicos en Learning Analytics, observamos que se suele usar Python combinado con R en este campo, por lo que se decidió usar estos lenguajes conjuntamente.

Python es un lenguaje de propósito general, orientado a objetos e independiente de plataforma. Sus principales características son:

- Sintaxis clara y escueta, lenguaje muy simple y de alto nivel. La característica principal que se puede resaltar de Python es su simplicidad. Además, Python es uno de los lenguajes con mayor información por volumen de código.
- Open Source, multiplataforma. Su naturaleza Open Source ha servido para motivar su adaptación a diversas plataformas.
- Lenguaje orientado a objetos y de scripting, permitiendo la ejecución de sentencias Python a partir de línea de comandos o desde otros lenguajes.
- Lenguaje interpretado, no necesita compilar para ejecutar un código Python, aunque tiene compilador por motivos de rendimiento.
- Librerías muy extensas y muy probadas debido a la importancia y al grado de uso del lenguaje desde los últimos años.

R es un lenguaje para el análisis estadístico y gráfico. Sus principales características son:

- Lenguaje interpretado y de scripting, permitiendo la ejecución de sentencias R a partir de línea de comandos.

- Open Source, multiplataforma.
- Lenguaje orientado al estudio de datos estadísticos.
- Gran capacidad a la hora de realizar gráficos.
- Librerías extensas, existen más de 4000 paquetes en CRAN.

Se ha optado por usar Python para el tratamiento de datos y generación de indicadores, ya que es capaz de trabajar muy bien con JSON, lenguaje en el que se representan los datos sobre los eventos que generan los estudiantes.

Una vez disponemos de los datos en nuestra base de datos, usaremos R tanto para aplicar a los datos técnicas de Machine Learning y modelos predictivos como para la representación de las estadísticas, ya que R dispone de muchas librerías dedicadas a este propósito.

A continuación destacaremos las principales librerías de R que se van a utilizar.

Caret (acrónimo de "Classification and Regression Training") es una librería de R que proporciona las principales funciones para la creación de modelos predictivos, así como las estadísticas que se pueden obtener a partir del mismo.

Shiny es una librería de R que permite crear fácilmente aplicaciones web interactivas, permitiendo interactuar con los datos sin tener que manipular el código. Se basa en la programación reactiva, que vincula los valores de entrada con los de la salida, de forma que variaciones en la entrada provocan modificaciones en la salida.

Plotly es una librería de R para la visualización de gráficas cuya principal característica es la interacción con el contenido de la gráfica.

2.2.3. Modelos predictivos

Tras una investigación sobre los distintos tipos de modelos predictivos, y teniendo en cuenta trabajos previos similares descritos en la introducción, como [5] o [9], se ha decidido utilizar una lista concreta de modelos predictivos. Además, se han escogido de forma que abarquen distintos enfoques de aprendizaje que se pueden aplicar.

Boosted Logistic Regression (LogitBoost)

Este modelo predictivo se ha escogido principalmente por el uso de la regresión logística, una de las técnicas más populares utilizadas a la hora de predecir variables binarias.

Este modelo utiliza la técnica de boosting, una técnica de normalización cuyo objetivo es reducir el sesgo y la varianza de los datos de entrada, y se basa en la obtención de un clasificador fuerte a partir de clasificadores débiles. En particular, utiliza AdaBoost (Adaptative Boosting), una variante que itera sobre los clasificadores débiles para lograr un clasificador fuerte más potente. Para entender mejor la técnica AdaBoost se recomienda el siguiente artículo de los propios desarrolladores del algoritmo [34].

Juntando ambas técnicas se obtiene el algoritmo Boosted Logistic Regression.

Este modelo predictivo se encuentra en la librería *caTools* de R, y para poder referenciarlo a la librería *caret* lo almacena bajo el identificador *LogitBoost*.

Stochastic Gradient Boosting (gbm)

Este modelo predictivo se ha escogido principalmente para ser comparado con el modelo anterior.

Al igual que en LogitBoost, este modelo utiliza una variante de la técnica AdaBoost, pero en este caso utiliza una técnica para añadir aleatoriedad a la hora de definir los clasificadores débiles. Para entender mejor la técnica se recomienda el siguiente artículo [35].

Este modelo predictivo se encuentra en la librería *gbm* de R, y para poder referenciarlo a la librería *caret* lo almacena bajo el identificador *gbm*.

Extreme Gradient Boosting (XGBoost)

Este modelo predictivo también se ha escogido para ser comparado con LogitBoost.

Este modelo también utiliza una variante de la técnica AdaBoost, y se diferencia de gbm en aspectos que tienen que ver con el control de sobreajuste de los parámetros asociados a los clasificadores débiles.

Este modelo predictivo se ha escogido además por ser uno de los más utilizados en las competiciones de Kaggle [36] ref, por lo que resulta aun más interesante tenerlo en cuenta a pesar de tener ya dos algoritmos similares.

Este modelo predictivo se encuentra en la librería *xgboost* de R, y para poder referenciarlo a la librería *caret* lo almacena bajo el identificador *xgbLinear*.

Support Vector Machine (SVM)

Este modelo predictivo se ha escogido para ver otros enfoques distintos al boosting. Las máquinas de soporte vectorial basan su funcionamiento en la búsqueda de un hiperplano separador en el espacio de características a partir de las denominadas funciones kernel.

La principal diferencia que tienen las SVM con otros métodos de aprendizaje es que, mientras la mayoría de estos métodos se centran en la minimización de los errores de entrenamiento, las SVM se centran en la minimización del denominado riesgo estructural. La reducción del riesgo estructural se basa en dos ideas básicas: la primera en utilizar los denominados vectores soporte, que son un subconjunto de vectores de entrenamiento con propiedades específicas; la segunda se basa en, a partir de estos vectores soporte, encontrar el hiperplano separador que maximice la distancia a la que se encuentra de los vectores soporte.

Para más detalles de las SVM se recomienda el siguiente artículo [37]

Este modelo predictivo se encuentra en la librería *kernlab* de R, y para poder referenciarlo a la librería *caret* lo almacena bajo el identificador *svmLinear*.

K-Nearest Neighbors

Este modelo predictivo se ha escogido para tener otro nuevo enfoque, además de ser un modelo básico en sistemas de recomendación.

Este modelo utiliza una técnica de predicción por similitud que consiste en colocar en el espacio de variables los vectores del entrenamiento. Para clasificar un vector nuevo se calculan los k vectores de entrenamiento a menor distancia y se clasifica en función de la clase que contenga más vectores de entre los k elegidos. A partir de este modelo básico, se pueden generar distintos modelos cambiando la función distancia y la forma de elegir la k.

Para este modelo se utilizará la distancia a la que el ser humano está más habituado, que es la distancia euclídea.

Este modelo predictivo se encuentra en la librería *kknn* de R, y para poder referenciarlo a la librería *caret* lo almacena bajo el identificador *kknn*.

3

Funcionalidad y Análisis de requisitos

3.1. Funcionalidad y Módulos

Tras haber realizado un estudio sobre el estado de arte, ya estamos preparados para afrontar el desarrollo de la aplicación en profundidad. En esta sección se presentan tanto los requisitos funcionales como los no funcionales de la aplicación a desarrollar en este proyecto. Es de gran importancia definirlos lo mejor posible para lograr una aplicación robusta con una funcionalidad acotada, para que pueda escalar bien cuando se use en el futuro para otros proyectos.

Se ha decidido dividir la aplicación en distintos módulos con una funcionalidad concreta, tal y como se muestra en la figura 3.1. A continuación se describe brevemente la funcionalidad de cada uno de ellos.

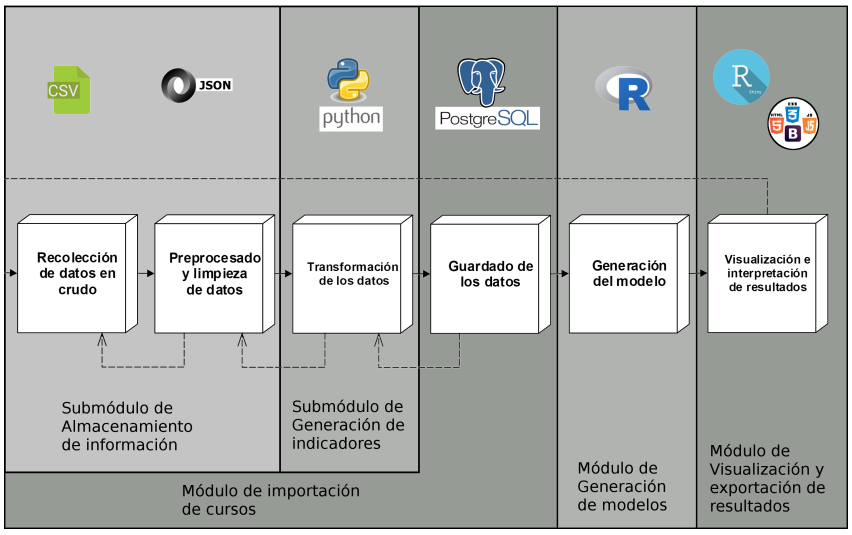

Figura 3.1: Módulos en los que se dividirá la aplicación

3.1.1. Módulo de Importación de cursos

Este módulo se encarga de todo el proceso de importación de datos de una edición de un curso MOOC en la base de datos, pasando por los distintos procesos de preprocesado, limpieza, transformación y almacenamiento de los datos. Este módulo se divide en dos submódulos, uno encargado del almacenamiento de datos que no necesitan transformaciones profundas y otro dedicado exclusivamente a la generación de indicadores.

3.1.2. Submódulo de Almacenamiento de información

Este módulo se encarga de almacenar todos los datos que se disponen de un curso MOOC en la base de datos, así como de almacenar todos los datos que se generarán a partir de ellos. Viendo el proyecto como un problema de Learning Analytics o de Data Science, este módulo se encargaría del preprocesado, limpieza y guardado de datos.

3.1.3. Submódulo de Generación de indicadores

Este módulo se encarga de generar distintos indicadores o métricas a partir de los ficheros de eventos de estudiantes de los cursos MOOC. Los indicadores que definiremos, en principio, asociarán un día de un curso y un estudiante con el correspondiente valor del indicador (a estos indicadores les llamaremos indicadores base). Puesto que nos interesa obtener indicadores aplicados a distintos intervalos de tiempo, este módulo también se encarga de obtener indicadores temporales a partir de los indicadores base.

En particular, en los modelos predictivos que se usarán tendrán como variables de entrada indicadores temporales acumulados, que, fijado un día del curso, el indicador acumulado resulta ser la suma de los indicadores base desde el primer día del curso hasta el día fijado. Por ejemplo, el indicador temporal acumulado *(course,user,day)* $\rightarrow$ *ind* resulta ser la suma de los indicadores base *(course,user,day)* $\rightarrow$ *ind* iterando el día desde el primer día hasta el día indicado.

La tabla 3.1 muestra la lista de indicadores diseñados. Véase 4.2.1 para una descripción más detallada de estos indicadores.

num_events	
num_sessions	total_time
nav_events	nav_time
video_events	video_time
forum_events	forum_time
problem_events	problem_time

Cuadro 3.1: Tabla de indicadores

Viendo el proyecto como un problema de Learning Analytics o de Data Science, este módulo se encargaría de la transformación de los datos.

3.1.4. Módulo de Generación de modelos

Este módulo se encarga de generar distintos modelos predictivos a partir de los datos transformados de la base de datos. Solamente habrá un tipo de modelo predictivo en función de lo que queremos predecir, el modelo predictivo de certificados, que se encargará de predecir si un estudiante aprueba el curso. Además habrá diferentes tipos de modelos predictivos en función de su algoritmo de aprendizaje.

En este caso no hace falta describir la funcionalidad de este módulo visto como un problema de Learning Analytics o de Data Science, pues es el nombre del propio módulo.

3.1.5. Módulo de Visualización y exportación de resultados

Este módulo se encarga tanto de la representación visual de la aplicación como de la comunicación entre la capa de representación con la lógica de todo el sistema. De forma más específica, este módulo se encarga de:

- Representar una interfaz gráfica para la importación de cursos a la base de datos así como la comunicación con los módulos encargados de esta tarea (el Módulo de Almacenamiento de información y el Módulo de Generación de indicadores).
- Representar una interfaz gráfica para la creación y guardado de modelos predictivos así como la comunicación con los módulos encargados de esta tarea (el Módulo de Almacenamiento de información y el Módulo de Generación de modelos).
- Representar una interfaz gráfica para la visualización de las estadísticas de un modelo predictivo, así como la selección del mejor modelo predictivo.
- Representar una interfaz gráfica para la exportación de los resultados de las estadísticas de un modelo predictivo.

Visto como un problema de Learning Analytics o de Data Science, este módulo (una parte de él en concreto) se encargaría de la visualización e interpretación de resultados. Por similitud se ha decidido incluir la visualización de toda la aplicación en este módulo.

3.2. Análisis de requisitos

A continuación se describen los requisitos funcionales que tendrá cada módulo, así como los requisitos no funcionales generales de la aplicación.

3.2.1. Requisitos funcionales: modelo de casos de uso

Para la descripción de requisitos funcionales vamos a utilizar el modelo de casos de uso, que describe las funciones que el sistema debe poder ser capaz de realizar para cada rol que interactuará con la aplicación. En principio solamente tendremos un único rol de usuario; se ha considerado incluir roles para diferenciar administradores e instructores, pero se han decidido descartar puesto que la aplicación final no va a tener un sistema de login para diferenciar estos roles.

Un caso de uso especifica una secuencia de acciones principales que realiza el sistema como respuesta a una interacción del usuario con la aplicación, incluyendo además distintos flujos alternativos a la secuencia principal.

Se definirá una lista de los distintos casos de uso de la aplicación en función de los distintos módulos descritos.

Módulo de Importación de cursos

RF-01: Importar datos de una edición de un curso	
Actor	Usuario
Descripción	El sistema deberá permitir al usuario importar todos los datos de una edición de un curso
Precondiciones	El usuario ha abierto la aplicación El usuario dispone de todos los datos de la edición del curso a importar, con el formato requerido que se describe a continuación.
Postcondiciones	El sistema almacena todos los datos de la edición del curso y muestra un mensaje de importación con éxito
Escenario principal	
1.	El usuario entra en la pestaña de importación de cursos y pulsa en “Folder select”, y selecciona el directorio que contiene todos los datos del curso. Es necesario que dentro de esta carpeta los eventos en formato JSON estén dentro del directorio “events” y el fichero CSV de certificados en el directorio “certificates”.
2.	El usuario introduce el nombre del curso y el nombre de la edición
3.	El usuario selecciona los indicadores que quiere guardar
4.	El usuario pulsa “Import course”
5.	El sistema comprueba si los datos para ese curso y esa edición existen, genera los indicadores seleccionados y guarda todos los datos preprocesados y transformados en la base de datos. Durante este proceso, el sistema muestra por pantalla el progreso de la operación.
6.	Una vez finalizado, el sistema muestra un mensaje de importación con éxito.
Flujos alternativos	
2'.	El usuario introduce los campos de curso y edición ya existentes en la base de datos.
3'.	El usuario selecciona los indicadores que quiere guardar.
4'.	El usuario pulsa “Import course”.
5'.	El sistema detecta que ese curso y esa edición ya ha sido importada. El sistema comprueba si alguno de los indicadores seleccionados ya están guardados.
5'.1	El sistema detecta varios indicadores ya guardados. El sistema muestra un mensaje con la lista de indicadores ya importados.
5'.1'	El sistema no detecta ningún indicador ya guardado. Continúa el flujo normal

Cuadro 3.2: Requisito funcional 1

RF-02: Generar y guardar un modelo predictivo para una edición de un curso	
Actor	Usuario
Descripción	El sistema deberá permitir al usuario generar un modelo predictivo asociado a una edición de un curso.
Precondiciones	El usuario ha abierto la aplicación El usuario ha importado la edición del curso en cuestión previamente.
Postcondiciones	El sistema almacena todos los datos del modelo y muestra un mensaje de importación con éxito.
Escenario principal	
1.	El usuario entra en la pestaña de modelos predictivos y selecciona el curso y la edición para la cual desea crear un modelo.
2.	El usuario selecciona los indicadores que quiere utilizar en el modelo predictivo, así como la variable de salida (solamente existe la de certificados) y el nombre del modelo predictivo.
3.	El usuario pulsa “Create model”
4.	El sistema comprueba si existe un modelo para los datos de entrada seleccionados, y genera el modelo predictivo. Durante el proceso, el sistema muestra por pantalla el progreso de la operación.
5.	Una vez finalizado, el sistema muestra un mensaje de tarea finalizada con éxito.
Flujos alternativos	
2'.	El usuario selecciona los campos de indicadores, variable de salida y modelo predictivo para los cuales ya existe un modelo en la base de datos asociado al curso y edición ya seleccionados.
3'.	El usuario pulsa “Create model”
4'.	El sistema detecta la existencia de un modelo para los datos de entrada seleccionados, y muestra por pantalla un mensaje de error indicando la existencia de dicho modelo.

Cuadro 3.3: Requisito funcional 2

Módulo de Generación de modelos

RF-03: Visualizar estadísticas de un modelo predictivo ya existente	
Actor	Usuario
Descripción	El sistema deberá permitir al usuario visualizar las estadísticas de un modelo predictivo.
Precondiciones	El usuario ha abierto la aplicación El sistema ya cuenta con un modelo guardado en la base de datos
Postcondiciones	El sistema muestra todas las estadísticas del modelo en cuestión.
Escenario principal	
1.	El usuario entra en la pestaña de modelos predictivos, selecciona la pestaña “Stats” y selecciona el curso y la edición para la cual desea visualizar el modelo.
2.	El usuario selecciona los indicadores que quiere utilizar en el modelo predictivo, así como la variable de salida (solamente existe la de certificados) y el nombre del modelo predictivo.
3.	El sistema muestra una pantalla con las diferentes estadísticas asociadas al modelo guardado con esas características seleccionadas. En particular, el sistema muestra: • Una pestaña en la que se muestran gráficas comparativas de los tiempos de entrenamiento, los tiempos de testeo y la medida AUC de todos los modelos guardados asociados a la edición del curso seleccionada. • Una pestaña en la que se muestra la gráfica de la evolución de las curvas ROC para el modelo particular seleccionado. • Una pestaña en la que se muestra la gráfica de comparativa de importancia de las variables para el modelo particular seleccionado. • Una pestaña en la que se muestra una tabla interactiva con los indicadores temporales acumulados de todos los usuarios del curso, para un día concreto.
Flujos alternativos	
2'.	El usuario selecciona los campos de indicadores, variable de salida y nombre de modelo predictivo de forma que no existe un modelo asociado a esos campos.
3'.	El sistema muestra las gráficas vacías.
3".	El usuario modifica alguno de los campos de las secuencias 1 y 2,
3".1	El sistema actualiza los datos de todas las gráficas con los nuevos datos de entrada.
4'"	El usuario selecciona en las gráficas asociadas a los tiempos y al AUC los modelos predictivos que quiere esconder/mostrar.
4'".1	El sistema actualiza las gráficas asociadas mostrando solamente las gráficas de los modelos seleccionados, ocultando las gráficas de los modelos no seleccionados.

Cuadro 3.4: Requisito funcional 3

Módulo de Visualización y exportación de resultados

RF-04: Seleccionar el mejor modelo predictivo asociado a una edición de un curso	
Actor	Usuario
Descripción	El sistema deberá permitir al usuario seleccionar el mejor modelo asociado a una edición de un curso.
Precondiciones	El usuario ha abierto la aplicación El sistema ya cuenta con un modelo guardado en la base de datos
Postcondiciones	El sistema muestra un mensaje con el mejor modelo actual
Escenario principal	
1.	El usuario entra en la pestaña de modelos predictivos, selecciona la pestaña "Stats" y selecciona el curso, la edición y los indicadores para determinar el mejor modelo asociado.
2.	El usuario selecciona la pestaña en la que se muestran los tiempos y la medida AUC.
3.	El sistema muestra, además de las gráficas comparativas, un mensaje con los el mejor modelo calculado por el sistema como sugerencia, así como el mejor modelo guardado en ese momento.
4.	El usuario pulsa el botón de guardar mejor modelo sugerido.
5.	El sistema actualiza el mejor modelo sugerido y actualiza el mensaje.
Flujos alternativos	
4'.	El usuario selecciona el modelo que quiere guardar como el mejor y pulsa el botón de guardar mejor modelo seleccionado.

Cuadro 3.5: Requisito funcional 4

RF-05: Exportar estadísticas de una edición de un curso	
Actor	Usuario
Descripción	El sistema deberá permitir al usuario exportar las estadísticas de un modelo predictivo.
Precondiciones	El usuario ha abierto la aplicación El sistema ya cuenta con un modelo guardado en la base de datos
Postcondiciones	El sistema muestra todas las estadísticas del modelo en cuestión.
Escenario principal	
1.	El usuario entra en la pestaña de modelos predictivos, selecciona la pestaña "Stats" y selecciona el curso y la edición para la cual desea realizar una exportación de gráficos.
2.	El usuario selecciona los indicadores que quiere utilizar en el modelo predictivo, así como la variable de salida (solamente existe la de certificados) y el nombre del modelo predictivo.
3.	El sistema una pantalla con las pestañas indicadas en el RF-03. Para cada gráfica o tabla seleccionada, el sistema dispondrá de un botón para exportar a CSV dicha gráfica o tabla, así como un selector de la carpeta destino donde se guardará..
4.	El usuario elige qué gráfica o tabla decide exportar a CSV, selecciona la carpeta de destino y pulsa el botón "Exportar"
5.	El sistema muestra un mensaje de éxito en la exportación.

Cuadro 3.6: Requisito funcional 5

Diagrama de casos de uso

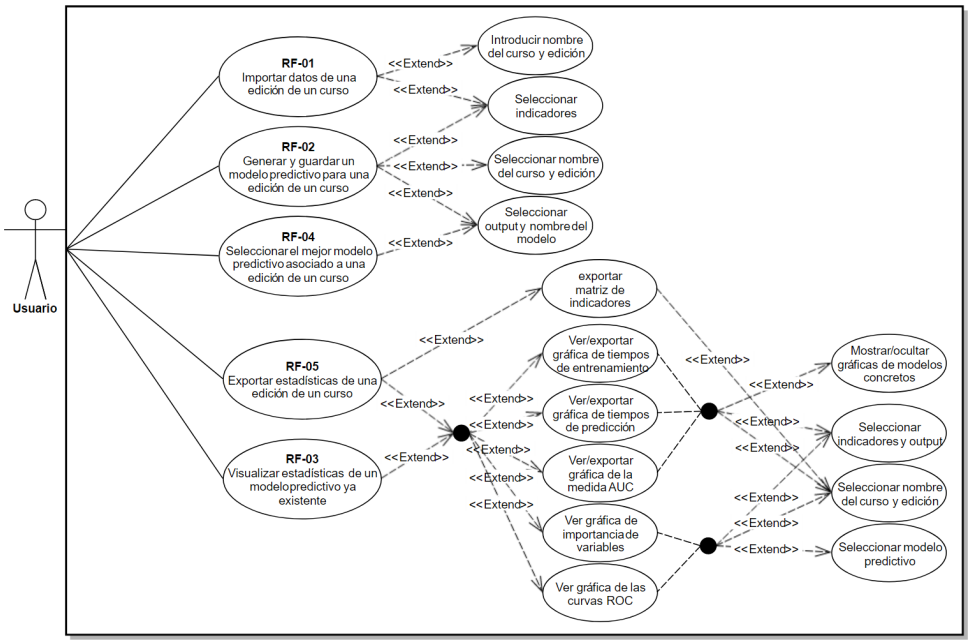

Figura 3.2: Diagrama de casos de uso completo de la aplicación

3.2.2. Requisitos no funcionales

RNF 1: Interfaz y usabilidad	
Descripción	La interfaz del sistema ha de ser clara e intuitiva. Se dispondrá de un mecanismo para evaluar la usabilidad, como por ejemplo una encuesta de usabilidad
Prioridad	Alta
RNF 2: Rendimiento	
Descripción	El sistema debe poder importar una edición de un curso en cuestión de minutos. No se impondrán restricciones temporales a la hora de generar modelos, pues depende de la propia naturaleza del modelo. El sistema debe actualizar las estadísticas de modelos generados en pocos segundos.
Prioridad	Baja
RNF 3: Seguridad	
Descripción	La base de datos deberá tener un usuario y contraseña conocidos solamente por los administradores. La base de datos deberá prevenir un ataque de SQL-Inyection.
Prioridad	Media
RNF 4: Documentación	
Descripción	Se deberá incluir un manual de usuario de la aplicación, que explique la ejecución de los distintos casos de uso y los diferentes flujos.
Prioridad	Alta
RNF 5: Escalabilidad	
Descripción	El sistema debe poder escalar fácilmente a la hora de añadir nuevos indicadores y nuevos modelos predictivos.
Prioridad	Alta
RNF 6: Privacidad de datos	
Descripción	El sistema debe preservar la total confidencialidad de los estudiantes, de los cuales solamente se deberá conocer su identificador.
Prioridad	Alta

Cuadro 3.7: Requisitos no funcionales

4

Diseño

El diseño de la aplicación se ha realizado teniendo en cuenta tanto los módulos definidos previamente como los requisitos funcionales de cada módulo. En primer lugar se explica la arquitectura lógica del sistema, para posteriormente detallar el diseño realizado por cada módulo.

4.1. Arquitectura lógica

La arquitectura lógica es un diseño de alto nivel de la estructura que va a tener una aplicación, y describe sus componentes principales y las relaciones que se establecen entre ellos.

Puesto que la aplicación a desarrollar está basada en una aplicación web, se ha elegido una arquitectura de tres capas: capa de presentacion, capa de negocio y capa de datos. A continuación se describe cada una de ellas. La figura 4.1 muestra el esquema de la arquitectura lógica.

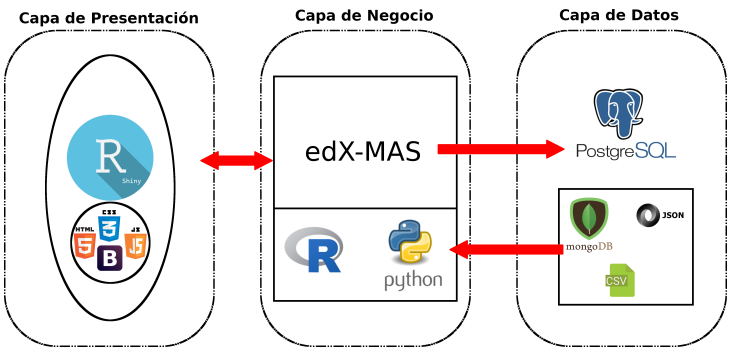

Figura 4.1: Arquitectura lógica del sistema

4.1.1. Capa de presentación

La capa de presentación es la que el usuario ve al usar la aplicación. Se encarga de presentar la información al usuario y de capturar la interacción del usuario con la interfaz gráfica. En esta capa se encuentran todos los ficheros de tipo cliente del framework Shiny, que internamente utiliza HTML5 para dar estructura y formato a la aplicación.

4.1.2. Capa de negocio

La capa de negocio se encarga de recibir las peticiones que realiza el usuario y de generar una respuesta tras el proceso. Se comunica con la capa de presentación tanto para recibir las peticiones como para enviar los resultados, y con la capa de datos para almacenar o extraer datos. Esta capa es la más importante del proyecto puesto que engloba los módulos principales de la aplicación, el Módulo de generación de indicadores y el Módulo de Generación de modelos. Además, esta capa contiene los ficheros que gestionan la conexión con la base de datos y los ficheros que se encargan de procesar las peticiones de los usuarios.

4.1.3. Capa de datos

La capa de datos se encarga de almacenar todos los datos relativos a los cursos MOOC. En esta capa se encuentra la base de datos y los distintos ficheros con los datos en crudo de los cursos MOOC. Además, se encarga de actualizar la base de datos via petición de la capa de negocio.

4.2. Diseño por módulos

A continuación se describe el diseño realizado por cada módulo, mostrando diagramas de diseño más relevantes para cada módulo.

4.2.1. Módulo de Importación de cursos

Submódulo de Almacenamiento de información

Puesto que la finalidad de este submódulo es almacenar todos los datos de un curso MOOC como los datos generados a partir de ellos, se describirá el diseño del modelo de datos de la aplicación, que se muestra en la figura 4.2.

- Tabla **course_runs**: Contiene la información básica para caracterizar una edición de un curso: nombre del curso y la edición. Podría ser interesante guardar otros campos como la universidad que imparte el curso, la url desde la que se puede acceder o las fechas de inicio o fin, pero para este proyecto no son relevantes.
- Tabla **course_duration**: Relaciona cada edición de un curso con su duración. Se decidió separar de la tabla course_runs para ser coherente con las siguientes

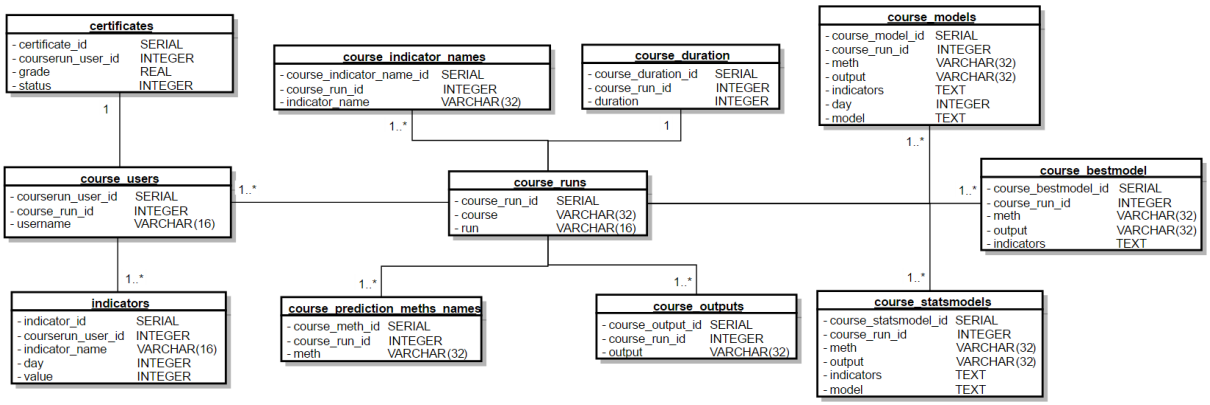

Figura 4.2: Modelo de datos

tablas que se describen y porque la tabla course_runs no tiene más campos. Sería recomendable colocar este campo en la tabla course_runs en caso de incorporar más campos a dicha tabla.

- Tabla **course_prediction_meths_names**: Relaciona cada edición de un curso con un método predictivo que se podrá aplicar sobre dicha edición. En un principio se podrá usar el mismo conjunto de métodos predictivos para cada edición de cada curso, pero se comprobó que alguno de estos métodos aplicados a cursos con muchos estudiantes fallaban a veces, lo que motivó a crear esta tabla.

- Tabla **course_outputs**: Relaciona cada edición de un curso con las posibles predicciones que se podrán aplicar a dicha edición. En principio se podrán predecir la medida descrita previamente para cada edición de un curso, pero para prevenir errores futuros se decidió crear esta tabla para poder filtrar las posibles métricas a predecir.

- Tabla **course_models**: Relaciona una edición de un curso, un método predictivo, una lista de indicadores que usará el modelo predictivo, una medida de salida a predecir y un día del curso con su variable de modelo asociado. Es decir, guarda el resultado de una función que transforma la variable model de R en una cadena de caracteres, de forma que se pueda volver a obtener la misma variable a partir de esa cadena de caracteres. Hay que tener en cuenta que el modelo predictivo asociado a un curso consta de tantos modelos como días lo conforma.

- Tabla **course_bestmodel**: Relaciona una edición de un curso con una referencia al mejor modelo seleccionado para esa edición. Este modelo queda determinado por sus atributos (lista de indicadores, variable de salida y nombre del método predictivo).

- Tabla **course_statsmodels**: Relaciona una edición de un curso, un método predictivo, una lista de indicadores que usará el modelo predictivo y una medida de salida a predecir con las estadísticas del modelo predictivo asociadas. Al igual que en la tabla anterior, se ha decidido guardar una transformación en texto plano de la variable en R que las contiene, de forma que a partir del texto plano se pueda recuperar dicha variable.

- Tabla **course_users**: Relaciona una edición de un curso con el username de un estudiante. Puesto que, como se ha comentado en la introducción, los datos de los estudiantes están anonimizados, el único dato que se guarda de un estudiante es su identificador. Se decidió no dividir esta tabla en una tabla con solo datos de estudiantes y una tabla relacional que relacionara estudiantes con ediciones de un curso debido a que solamente hay un campo para los estudiantes (su identificador), y porque es mucho más sencillo de insertar. Si en el futuro fuera necesario, sería recomendable dividir esta tabla en las dos descritas previamente.

- Tabla **certificates**: Relaciona el usuario de un curso con su nota final. En particular, guarda el la nota en la columna *grade* y su resultado final en la columna status (0 si suspende, 1 si aprueba).

- Tabla **indicators**: Relaciona un usuario de un curso, un nombre de indicador y un día del curso con el valor de dicho indicador.

Estas tablas serán pobladas por dos vías distintas. Todas las tablas salvo *course_models* y *course_statsmodels* se poblarán en las fases de preprocesado, limpieza, transformación y guardado de los datos, mientras que las otras dos tablas mencionadas se poblarán una vez se disponga de un modelo y de sus estadísticas asociadas.

Submódulo de Generación de indicadores

Para este submódulo mostraremos el diseño que se ha realizado para generar indicadores a partir de los ficheros de eventos. Realizar un buen diseño de los indicadores es de gran importancia, pues permite que la aplicación sea escalable a la hora de crear nuevos indicadores. En la figura 4.3 se muestra el diagrama de clases diseñado para los indicadores, que explicaremos a continuación.

Se parte del indicador base **EdxIndicator**, que contiene las funciones básicas para obtener todos los indicadores de todos los estudiantes asociados a un día de un curso concreto. Los métodos más importantes de esta clase son los siguientes:

- El método *getDayEvents* devuelve la lista de todos los eventos de un día de un curso.

- El método *getDayEventsPartition* realiza una partición de la lista anterior utilizando a los usuarios como representantes de las distintas clases de equivalencia de la partición. De esta forma, devolverá una lista de pares usuario - lista de eventos de dicho usuario.

- El método más importante de esta clase es *getIndicator*, que, a partir de todos los eventos de un usuario, devolverá un valor que será el valor del indicador. Este método será sobreescrito en las clases hijas para obtener el indicador correspondiente.

Para crear un nuevo indicador simplemente hay que crear una clase que herede del indicador base, y sobreescribir el método getIndicator. A pesar de ello el método puede resultar bastante complejo, por lo que se optó por la creación de más clases genericas a partir del indicador base.

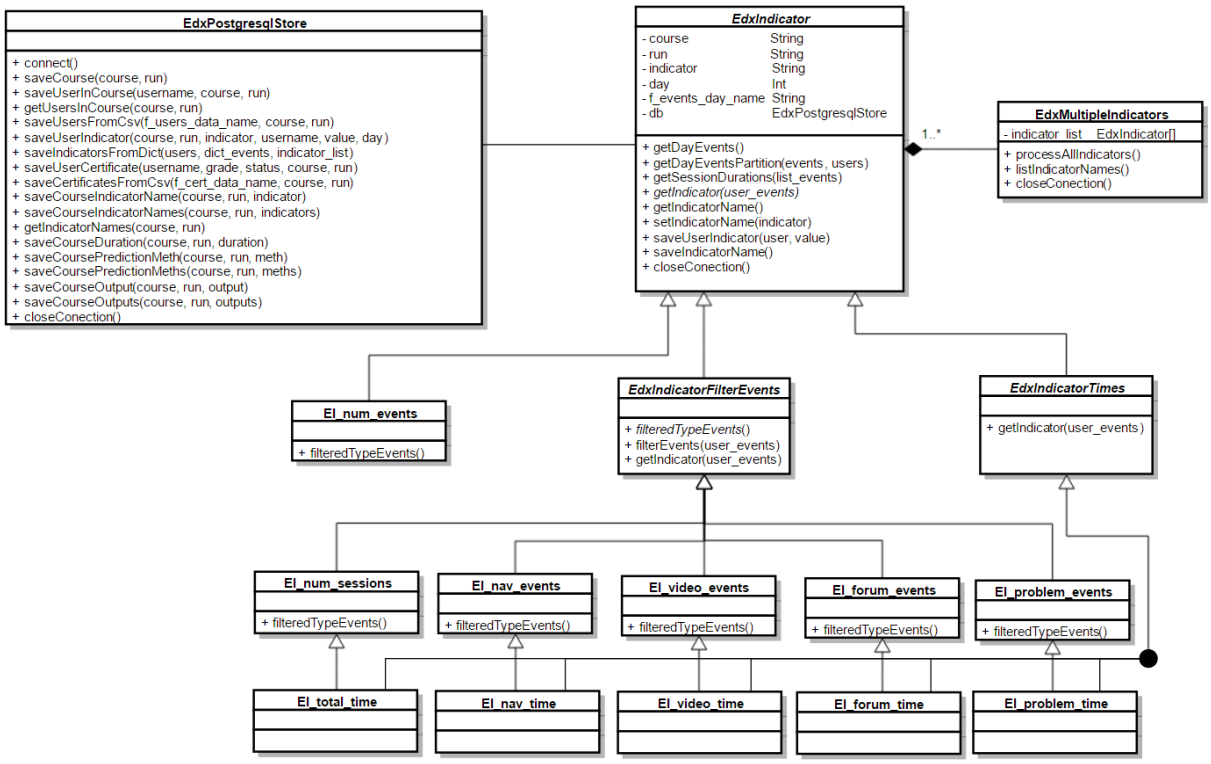

Figura 4.3: Diagrama de clases de los indicadores

La clase *EdxIndicatorFilterEvents* tiene como objetivo filtrar los eventos asociados a un usuario con respecto al tipo de evento, y generar como indicador la longitud de la lista de eventos tras el filtrado. De esta forma, para la creación de un indicador que cuente eventos de cierto tipo, basta con heredar de esta clase y sobreescribir el método *filteredTypeEvents*, que devuelve una lista con los nombres de los tipos de eventos que se quieren filtrar. A partir de esta clase se han creado los siguientes indicadores:

- *EI_num_sessions*: Este indicador devuelve el número de sesiones que ha tenido el estudiante en ese día. Este es el indicador menos parecido de esta lista pues necesita sobreescribir también el método *getIndicator*, y en principio podría heredar directamente del indicador base.
- *EI_nav_events*: Este indicador devuelve el número de eventos relacionados con la navegación entre los distintos contenidos de un curso (cambiar de capítulo, pinchar en un enlace, etc).
- *EI_video_events*: Este indicador devuelve el número de eventos relacionados con la visualización de vídeos (empezar a ver un vídeo, pausarlo, avanzarlo, etc)
- *EI_forum_events*: Este indicador devuelve el número de eventos relacionados con los foros del curso (crear posts, comentar en el post de otro estudiante, buscar, votar, etc)
- *EI_problem_events*: Este indicador devuelve el número de eventos relacionados con los problemas de un curso (seleccionar respuesta, escribir respuesta, guardar resultados de un problema, etc).

La clase *EdxIndicatorTimes* tiene como objetivo contar el tiempo que ha transcurrido a partir de una lista de eventos. Puesto que un estudiante ha podido entrar varias veces en un curso a lo largo del día, es necesario no considerar el tiempo en el que el estudiante ha estado desconectado. Para ello se ha creado en el indicador base el método *getSessionDurations*, que a partir de una lista de eventos, genera una partición de esa lista cuyas clases de equivalencia serán sublistas de eventos que están próximos temporalmente (como se ha comentado previamente, se han escogido de forma que sea menor de 5 minutos). A partir de esta partición se puede calcular fácilmente la duración de cada sublista, y esto último es lo que devuelve el método. Este método es utilizado también en la clase *EI_num_sessions*, ya que si le pasamos todos los eventos de un usuario a este método obtendríamos las sesiones del usuario. Se observa que, puesto que los indicadores creados previamente filtran eventos, se puede usar herencia múltiple para calcular los tiempos asociados a los eventos que pasan dicho filtro. Con esta filosofía se han creado los siguientes indicadores:

- *EI_total_time*: Este indicador devuelve el tiempo total estimado que el estudiante ha estado conectado.
- *EI_nav_time*: Este indicador devuelve el tiempo estimado que el estudiante ha estado navegando por los contenidos de un curso.
- *EI_video_time*: Este indicador devuelve el tiempo estimado que el estudiante ha estado viendo vídeos.
- *EI_forum_time*: Este indicador devuelve el tiempo estimado que el estudiante ha estado en el foro.
- *EI_problem_time*: Este indicador devuelve el tiempo estimado que el estudiante ha empleado resolviendo los problemas del curso.

Además de todas las clases derivadas del indicador base, se han creado otras dos clases para facilitar el proceso de obtención de indicadores. La clase *EdxPostgresqlStore* contiene todas las funciones necesarias para guardar todos los datos necesarios en la base de datos. En el caso de los indicadores, una vez obtenido un indicador, se usará esta clase para almacenarlo en la base de datos. Por otra parte, la clase *EdxMultipleIndicators* sirve para agilizar todo el proceso de obtención de indicadores de todo un curso, simplificando todo este proceso en el método *processAllindicators()*. Esta clase recibe en el constructor una lista con los indicadores que se quieren utilizar, y el método genera y guarda todos los indicadores obtenidos del curso en cuestión.

4.2.2. Módulo de Generación de modelos

Para este submódulo mostraremos el diseño que se ha realizado para generar los distintos modelos a partir de los datos guardados en la base de datos. Para este submódulo también resulta de gran importancia el diseño para permitir escalabilidad a la hora de introducir nuevos modelos predictivos. En la figura 4.4 se muestra el diagrama de clases diseñado para la generación de modelos, que explicaremos a continuación.

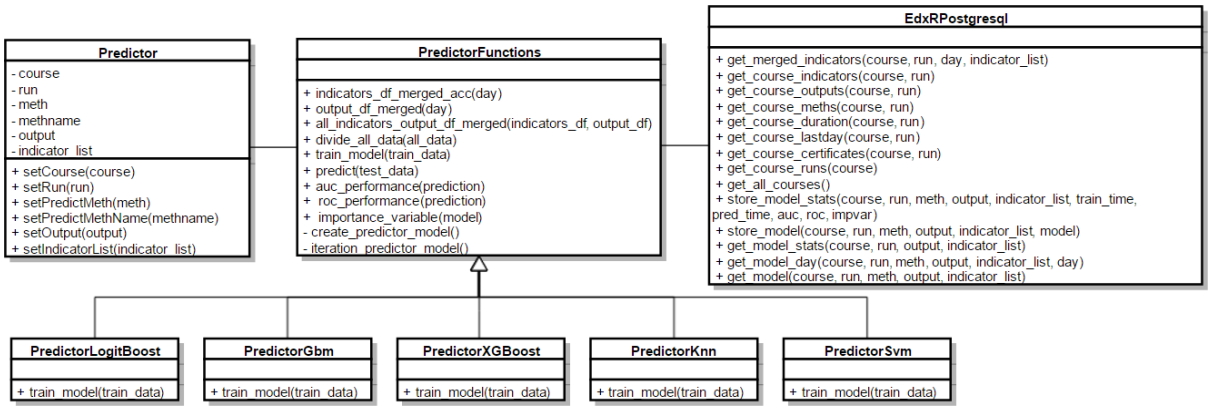

Figura 4.4: Diagrama de clases para la generación de modelos

Se parte de la clase base **PredictorFunctions**, que se construye a partir de una clase que contiene los datos que identifican al modelo y de otra clase dedicada a la gestión de la base de datos. El objetivo principal de la clase PredictorFunctions es definir una lista de métodos básicos que se llamarán por orden para obtener el modelo. De esta forma, a la hora de crear un predictor concreto, se podrán reutilizar la mayoría de métodos, y solamente haría falta sobreescribir alguno de ellos. A continuación describiremos los distintos métodos de esta clase.

- Lo primero que se necesita para generar el modelo es recuperar de la base de datos los indicadores acumulados asociados a un día de curso, es decir, la suma de indicadores base desde el primer día de curso hasta el día para el cual se quiere generar el modelo. Esta funcionalidad la cubre el método *indicators_df_merged_acc*, que devuelve una matriz cuyas primera columna es el identificador del estudiante, y el resto de columnas son los indicadores, y que tiene por filas las distintas tuplas usuario, valor del primer indicador, valor del segundo indicador, etc.

- Lo siguiente que se necesita es recuperar de la base de datos la variable de salida asociada a cada estudiante. Para este proyecto solamente se usa la de certificados, aunque la herencia y sobreescritura permitiría ser usada para otras variables de salida. Esta funcionalidad la cubre el método *output_df_merged*, que devuelve una matriz cuya primera columna es el identificador del estudiante y la segunda es el valor de la variable output asociada al estudiante, que en este caso será un valor booleano que indica si ha pasado el curso o no.

- A partir de ambas matrices, se necesita obtener una matriz que combine ambas matrices, y de esto se encarga el método *all_indicators_output_df_merged*. En este punto ya tendremos lo que denominaremos matriz de patrones, cada uno de los cuales está formado por un vector de variables de entrada (los indicadores) y otro de salida (el certificado).

- Ahora se creará una partición de los patrones, que a partir de la matriz anterior devuelva dos matrices con las mismas columnas. Una de ellas se utilizará a la hora de entrenar el modelo y la otra a la hora de testear el modelo. De esto se encarga el método *divide_all_data*

- A partir de la matriz de training se generará el modelo predictivo en cuestión, además de medir el tiempo que se tarda en generarlo. Esta funcionalidad la recoge el método *train_model*.
- A partir del modelo se realizará el testing y se sacarán las distintas estadísticas asociadas al modelo. El método *predict* devuelve el tiempo empleado en testear y una estructura a partir de la cual se pueden obtener las distintas estadísticas. Esta estructura se le pasará a los métodos *roc_performance*, *auc_performance* e *importance_variable*, que devuelven, respectivamente, una matriz de puntos para pintar la curva ROC, la medida AUC y una matriz con la importancia de las variables.
- Por último, el método *create_predictor_model* integra y utiliza los métodos descritos anteriormente para generar el modelo a partir de los datos del constructor.

4.2.3. Módulo de Visualización y exportación de resultados

Para este submódulo se ha propuesto definir el diseño que tendrá la aplicación a la hora de interactuar con ella, y cómo se definen el flujo de información entre la interacción con la interfaz gráfica y el resto de partes de la aplicación. Para ello se realizará un diseño MVC para todos los componentes interactivos de la aplicación. Por otra parte se diseñará la estructura que tendrán estos componentes interactivos.

El flujo de datos se realizará con el patrón MVC y se resume en la figura 4.5. El patrón MVC básicamente resuelve el problema de flujo de información de la interfaz gráfica a las funciones internas a través de un controlador, que se encarga de realizar la correcta comunicación, de manera que ambas partes se puedan desarrollar de forma independiente.

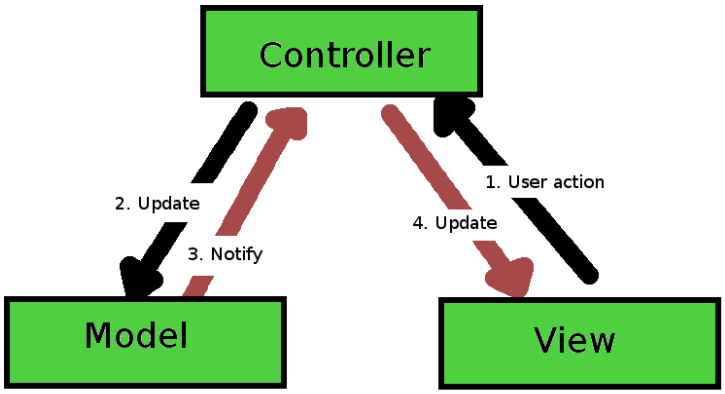

Figura 4.5: Patrón MVC

Para el diseño de la estructura de cómo se colocan los elementos interactivos en la interfaz gráfica, se han diseñado distintos mockups que no se muestran en este documento, pues ya se incluyen pantallazos de la interfaz real en el Anexo B.

5

Desarrollo e implementación

La fase de desarrollo del software se ha realizado teniendo en cuenta tanto el diseño como los módulos definidos previamente y sus requisitos funcionales. En este capítulo se vuelve a recordar el software que será usado para desarrollar la aplicación, y se explica la estructura de ficheros que tendrá la aplicación y los propios ficheros que la conforman.

Por último, se explicará la implementación abordándolo desde un contexto más práctico, describiendo las modificaciones que habría que realizar si se desea ampliar la aplicación.

5.1. Estructura de ficheros

La estructura de ficheros de la aplicación se resumen en la figura 5.1, que explicaremos más en profundidad a continuación.

```
./postgresqlstore
./postgresqlstore/createdatabase.sql
./postgresqlstore/edxindicatorstore.py
./postgresqlstore/edxpostgresqlstore.py
./postgresqlstore/script_all_course.py

./R
./R/edxrpostgresqldb.R
./R/global.R
./R/predictor.R
./R/server.R
./R/src
./R/ui.R

./R/src/server
./R/src/server/courseimporter.R
./R/src/server/interactiveUI.R
./R/src/server/modelcreator.R
./R/src/server/statsgraphics.R

./R/src/ui
./R/src/ui/importcourses.R
./R/src/ui/predictivemodel.R
```

Figura 5.1: Estructura de ficheros de la aplicación

La aplicación se divide en dos directorios claramente diferenciados por su funcionalidad, que son el directorio *postgresqlstore* y el directorio *R*.

En el directorio *postgresqlstore* se encuentran los ficheros que desempeñan la funcionalidad del módulo de Importación de cursos. Consta de 3 ficheros tipo Python y uno

escrito en SQL.

- El fichero *createdatabase.sql* es el encargado de crear la estructura de la base de datos basada en el modelo descrito en el diseño.
- El fichero *edxindicatorstore.py* contiene las clases y los métodos para la generación de indicadores.
- El fichero *edxpostgresqlstore.py* contiene una clase que se encarga de la conexión y de la gestión de la base de datos a la hora de importar cursos, por lo que en general gestionará consultas de tipo INSERT.
- El fichero *script_ all_ course.py* contiene un main que, a partir de una url en la que se encuentran los datos del curso, el nombre del curso, el nombre de la edición, y una lista de argumentos de nombres de indicadores, guarda todos los datos del curso y genera y guarda los indicadores pasados por argumentos, utilizando las clases y métodos desarrollados en los dos ficheros anteriores.

En el directorio *R* se encuentran el resto de ficheros de la aplicación, todos ellos escritos en R. Estos ficheros se dividen según su funcionalidad, unos desempeñan la funcionalidad del Módulo de Generación de modelos y otros la funcionalidad del Módulo de Visualización y exportación de resultados.

Los ficheros que tienen que ver con el módulo de Generación de modelos son los siguientes:

- El fichero *predictor.R* contiene las clases y los métodos para la generación de modelos.
- El fichero *edxpostgresqldb.R* contiene una clase que se encarga de la conexión y de la gestión de la base de datos a la hora de generar modelos, por lo que en general gestionará consultas de tipo SELECT.

Los ficheros que tienen que ver con el Módulo de Visualización y exportación de resultados son los siguientes:

- Los ficheros *server.R*, *ui.R* y *global.R* son ficheros base utilizados por la librería Shiny para la creacíon de una aplicación tipo Shiny. El fichero *ui.R* se encarga de definir la interfaz de usuario, mientras que el fichero *server.R* sirve de controlador y controla la respuesta de la aplicación cuando el usuario interactúa con la aplicación. Por último, el fichero *global.R* contiene las funciones globales que conectarán la capa server de Shiny con el resto de ficheros que se han desarrollado. Para que los ficheros server.R y ui.R no queden tan extensos, se han subdividido cada uno en ficheros situados en los directorios *R/src/server* y *R/src/ui*.

5.2. Implementación y escalabilidad

A continuación presentamos dos escenarios posibles de ampliación de la aplicación que se consideran bastante importantes. y qué sería necesario modificar para incluir estas ampliaciones.

Crear un nuevo tipo de indicador. Si se quiere crear un nuevo tipo de indicador y que se use en la aplicación, se deben modificar los siguientes ficheros:

- En el fichero *edxindicatorstore.py* se debe crear una nueva clase que herede de la clase *EdxIndicatorFilterEvents* en caso de ser un indicador que cuente eventos de un conjunto de tipos de eventos. Bastaría con redefinir la función *filteredTypeEvents* para que devuelva una lista con los tipos de indicador que se quieren filtrar.

  En caso de querer medir el tiempo entre eventos del mismo conjunto de tipos de eventos, se deberá crear otra nueva clase que herede de las clases *EdxIndicatorTimes* y la clase anterior. Bastaría con definir la clase, puesto que con la herencia múltiple la clase ya tendría todos los métodos.

  En caso de querer generar otro tipo de generador que no tiene nada que ver con lo descrito anteriormente, habría que crear una clase que heredase directamente de *EdxIndicator* y redefinir el método *getIndicator*.

- En el fichero *script_ all_ course.py* habría que añadir al diccionario que asocia parámetros de entrada con clases Python el nombre que se va a usar para identificar el indicador.

- Para que el nuevo indicador salga en la interfaz gráfica, se debe modificar la función *get_ default_ indicators* del fichero *global.R* y añadir al retorno el nombre del indicador.

Crear un nuevo modelo predictivo. Si se quiere crear un nuevo modelo predictivo habrá que modificar los siguientes ficheros:

- En el fichero *global.R* hay que modificar la función *get_ course_ meths* que añada el nombre del método predictivo que va a mostrar para dicho modelo. Además hay que modificar la variable *methdict*, un diccionario que tiene como clave el nombre del método que se va a mostrar por pantalla, y como valor un array con el nombre del método predictivo de la librería *caret* y la clase de tipo *predictorFunctions*.

- En principio bastaría con esta modificación para añadir un modelo predictivo de la librería *caret*. De todas formas puede que se quisiera crear un modelo predictivo que no estuviera incluido en dicha librería, o que se quiera un mayor grado de personalización. Para ello habría que añadir una nueva clase en el fichero *predictor.R* que herede de *predictorFunctions* y sobreescribir los métodos necesarios. Habría otra opción bastante interesante que consistiría en adaptar el modelo predictivo para que pueda ser procesado por la librería *caret*, y de esta forma la comunidad de R podría disfrutar del modelo predictivo creado.

6 Pruebas

En este capítulo se describe el conjunto de pruebas que se han realizado en el sistema. Para la realización de las pruebas se han usado datos reales de los MOOCs que la UAM hospeda en edX.

A continuación se explican los distintos tipos de pruebas que se han realizado sobre el sistema.

6.1. Pruebas unitarias

El objetivo de las pruebas unitarias es controlar y verificar la correcta funcionalidad de cada módulo independientemente del resto de la aplicación. Este proceso de pruebas se ha realizado de forma individual sobre cada fichero de la aplicación. A continuación se resume en qué ha consistido este proceso de pruebas unitarias dividido en los distintos módulos de la aplicación.

- Para el Módulo de Importación de cursos se han realizado pruebas unitarias sobre los distintos métodos de la clase encargada de la conexión y la gestión de la base de datos, comprobando que cada método funciona correctamente. Además, se ha probado de forma independiente de la base de datos la correcta generación de todos los indicadores desarrollados, comprobando la correcta salida con los datos de entrada para cada tipo de indicador.

- Para el Módulo de Generación de modelos se ha realizado una estrategia similar al anterior módulo. Por una parte se ha probado que los métodos relacionados con el acceso a la base de datos devuelven los resultados en el formato deseado y de forma correcta. Por otro lado, se ha comprobado que cada método relacionado con la generación del modelo devuelve de forma correcta lo esperado a partir de los argumentos de entrada. Por último, se ha comprobado para cada modelo predictivo

particular si el paso de argumentos ha sido correcto y si el resultado obtenido era el esperado.

- Para el Módulo de Visualización y exportación de estadísticas, las pruebas unitarias que se han realizado son ligeramente distintas a las anteriores, pues aquí se encuentran todos los ficheros relacionados con la aplicación web. Por un lado se ha probado que los métodos encargados de mostrar la interfaz gráfica funcionan correctamente y muestran lo esperado. Por otro lado se han probado de forma independiente el correcto funcionamiento de las funciones que devuelven las matrices de los datos a visualizar como de los datos a exportar.

6.2. Pruebas de integración

El objetivo de las pruebas de integración es comprobar que el flujo de información entre cada capa es el correcto. A continuación se resumen las pruebas de integración realizadas

- Tanto para el Módulo de Importación de cursos como para el Módulo de Generación de modelos se ha probado la propia integración interna, la cual ha sido sencilla de probar al ser un diseño pensado en el procesamiento por lotes, por lo cual solamente ha sido necesario comprobar el correcto funcionamiento de los métodos principales encargados de llamar al resto. Para el Módulo de Importación de cursos, se ha probado el correcto funcionamiento del script del fichero *script_all_courses.py*. Para el Módulo de Generación de modelos se ha probado el correcto funcionamiento del método *create_predictor_model*, que integra el resto de métodos.

- También se ha probado, para los mismos módulos, la correcta integración de los métodos relacionados con la conexión y gestión de la base de datos con los métodos tanto de la importación de cursos como de la generación de modelos.

- Para el Módulo de Visualización y exportación de estadísticas se han realizado distintos tipos de pruebas de integración. Por un lado se ha probado que todos los elementos interactivos de la interfaz gráfica están conectados con su controlador correspondiente. Por otro lado, se ha comprobado que los controladores llaman de forma correcta al resto de funcionalidades de la aplicación.

6.3. Pruebas de sistema

El objetivo de estas pruebas es comprobar que la aplicación funciona al ser desplegada en un entorno controlado por el desarrollador, y pretende encontrar limitaciones de hardware y/o software que no han sido encontradas en las anteriores pruebas. Para estas pruebas se ha desplegado la aplicación en dos entornos bastante diferenciados en cuanto a hardware y software. El primer entorno fue un entorno Linux sobre un ordenador portatil comercial relativamente nuevo y con SSD, mientras que el segundo entorno fue un entorno Windows sobre un ordenador sobremesa no tan nuevo y con HDD. Las pruebas realizadas han consistido en el correcto funcionamiento en ambos entornos y en estudiar si el rendimiento sobre el ordenador con un hardware más antiguo ha sido el adecuado.

6.4. Pruebas de validación

El objetivo de las pruebas de validación es comprobar que se cumplen todos los requisitos especificados. Para ello se ha comprobado que la aplicación cumple con los 5 requisitos funcionales definidos en 3

6.5. Pruebas de aceptación

El objetivo de estas pruebas es comprobar que el producto resultante es el deseado. Para ello se han realizado distintas reuniones en las que personal docente investigador de la UAM interesado en analizar datos de los MOOCs probaba la aplicación hasta conseguir la aprobación. Como complemento se ha creado una encuesta de usabilidad utilizando la herramienta de formularios de Google, que se detalla en el Anexo C.

7

Conclusiones y trabajo futuro

En este último capítulo se exponen las conclusiones más importantes que se han logrado a la hora de realizar este proyecto, así como los posibles trabajos futuros que se podrán realizar teniendo como base este proyecto.

7.1. Conclusiones

Desde la incorporación de la Universidad Autónoma de Madrid al consorcio edX [4] en 2015 han surgido distintas necesidades sobre el estudio de los cursos MOOC, que han dado lugar al estudio y realización de diferentes proyectos que han ido cubriéndolas poco a poco.

Como punto de partida de estos proyectos, se ha diseñado y desarrollado un sistema que pretende facilitar la aplicación de técnicas de Learning Analytics para la predicción de aprobados de un curso MOOC, integrando de manera sencilla en una misma aplicación todos la secuencia de pasos de un proyecto de Data Science, desde la carga, preprocesado, limpieza y transformación de los datos hasta la generación de modelos predictivos y el estudio de su calidad.

Para el desarrollo de este proyecto se han tenido que completar todas las metas definidas en el capítulo de introducción, y que se exponen a continuación.

Se ha realizado un exhaustivo análisis de los paquetes de datos de los cursos MOOC facilitados por la plataforma edX, de los cuales se ha conseguido separar la información relevante.

Se ha realizado un estudio general sobre aprendizaje electrónico, sobre los cursos MOOC y las analíticas de aprendizaje, así como la tecnología utilizada en estos campos a la hora de aplicar técnicas de Learning Analytics.

Se han diseñado y desarrollado una aplicación modular y escalable que permite:

- Importar una edición de un curso MOOC seleccionando los indicadores que se desean guardar.
- Generar distintos modelos predictivos para predecir si un alumno va a aprobar el curso.
- Visualizar distintas gráficas con los resultados de la calidad de los modelos predictivos.
- Exportar estos resultados para ser utilizados en otros análisis.

Se ha probado el sistema en un entorno real con conjuntos de datos reales, con cursos de miles de estudiantes y una grandísima cantidad de eventos generados por cada día de curso.

7.2. Trabajo futuro

Como se ha mencionado a lo largo del proyecto, uno de los rasgos que más se han tenido en cuenta es la escalabilidad, permitiendo adaptar y ampliar la herramienta desarrollada.

Se pueden seguir distintas líneas de desarrollo para ampliar la aplicación en distintos trabajos futuros.

Por un lado está todo lo relacionado con la obtención de nuevos indicadores. Se podrían crear otros tipos de indicadores genéricos diarios, como, por ejemplo, que devuelvan el porcentaje de actividad para cierto conjunto de eventos, u otros que no sean diarios si no relativos a todo el curso. También se podría investigar el diseño de indicadores personalizados a un curso en cuestión, para de esta forma intentar mejorar el contenido del propio curso (por ejemplo, analizando la importancia que han tenido distintos problemas, vídeos, etc).

Por otro lado se puede ampliar la generación de modelos, incluyendo metodologías de aprendizaje que no se han tenido en cuenta en este proyecto. Además sería muy interesante ampliar esta parte personalizando cada método predictivo, de forma que se le pudieran pasar parámetros personalizados que influyen en la predicción.

Por último, esta herramienta puede ser el punto de partida de un sistema de alarmas que alerte a los usuarios que no han abandonado y que están en riesgo de no aprobar el curso, así como recomendar sugerencias para salir de esa zona crítica utilizando el significado de los indicadores y su importancia en el modelo predictivo.

Bibliografía

[1] Moocs. *En línea. Disponible:* *https://es.wikipedia.org/wiki/Massive_Open_Online_Course*, [Último acceso: mayo 2017].

[2] Learning analytics. *En línea. Disponible:* *https://es.wikipedia.org/wiki/Learning_analytics*, [Último acceso: mayo 2017].

[3] Universidad autónoma de madrid. *En línea. Disponible:* *http://www.uam.es/*, [Último acceso: mayo 2017].

[4] edx. *En línea. Disponible:* *https://www.edx.org/*, [Último acceso: mayo 2017].

[5] R. Cobos, A. Wilde, and Zaluska. Predicting attrition from Massive Open Online Courses in FutureLearn and ed. Comparing attrition prediction in FutureLearn and edX MOOCs. *Proceedings of the LAK FutureLearn Worshop in the Learning Analytics and Knowledge 2017 Conference (LAK17), Canada.*, 13-17 Mar 2017.

[6] I.D. Claros, R. Cobos, G. Sandoval, and M. Villanueva. Creating MOOCs by UAMx: experiences and expectations. *European MOOC Stakeholder Summit 2015. Mons, Bélgica.*, 18-20 Mayo, 2015.

[7] I.D. Claros, L. Echeverría, A. Garmendia, and R. Cobos. Towards a Collaborative Pedagogical Model in MOOCs. *Global Engineering Education Conference (EDUCON), 2014 IEEE (EDUCON 2014). Estambul, Turquía.*, 3-5 Abril, 2014.

[8] M Leon, R. Cobos, and K. Dickens. Internal Perspectives of MOOCs in Universities. *European MOOC Stakeholder Summit 2017 (eMOOCs 2017). Leganés, Madrid, España.*, 22-26 Mayo, 2017.

[9] L.A. Ruipérez Valiente, R. Cobos, P.J. Muñoz-Merino, A.l Andújar, and C. Delgado Kloos. Early Prediction and Variable Importance of Certificate Accomplishment. *European MOOC Stakeholder Summit 2017 (eMOOCs 2017). Leganés, Madrid, España.*, 22-26 Mayo, 2017.

[10] M.l Leon, R. Cobos, K. Dickens, Su. White, and H. Davis. Visualising the MOOC experience: a dynamic MOOC dashboard built through institutional collaboration. *eMOOCs 2016. 4th European MOOCs Stakeholders Summit.*, pages 461–470, Feb 22-24, 2016.

[11] C. Delgado Kloos, C. Alario-Hoyos, C. Fernández-Panadero, I. Estévez-Ayres, P. Muñoz-Merino, R. Cobos, J. Moreno, E. Tovar, R. Cabedo, N. Piedra, J. Chicaiza, and J. López. Proyecto eMadrid: MOOCs y Analítica del Aprendizaje. SIIE16, CEDI2016. Salamanca, Castilla y León, España. 2016.

[12] R. Cobos, S. Gil, A. Lareo, and F.A. Vargas. Open-DLAs: An Open Dashboard for Learning Analytics. *LS 2016 Third (2016) ACM Conference on Learning Scale Edinburgh, Scotland Uk*, pages 265–268, April 25 - 26, 2016.

[13] S. Zamora Castro, A. Molina Navarro, M. Díaz Vega, and E. Lagunes Lagunes. Aprendizaje Electrónico: Puertas abiertas a la educación autodidacta a distancia. *Revista de Tecnologías de la Información*, 2016.

[14] Thecompleteuniversityguide. *En línea. Disponible:* `https://www.thecompleteuniversityguide.co.uk/distance-learning/moocs-(massive-open-online-courses)/`, [Último acceso: mayo 2017].

[15] Coursera. *En línea. Disponible:* `https://www.coursera.org/`, [Último acceso: mayo 2017].

[16] Futurelearn. *En línea. Disponible:* `https://www.futurelearn.com/`, [Último acceso: mayo 2017].

[17] Stanford university. *En línea. Disponible* `https://www.stanford.edu/`, [Último acceso: mayo 2017].

[18] Data science. *En línea. Disponible:* `https://es.wikipedia.org/wiki/Ciencia_de_datos`, [Último acceso: mayo 2017].

[19] M. Ballesteros. Data Analytics for MOOC Providers. *Trabajo fin de master, Universidad de Southampton*, 2016.

[20] F1score. *En línea. Disponible:* `https://en.wikipedia.org/wiki/F1_score`, [Último acceso: mayo 2017].

[21] Accuracy and precision. *En línea. Disponible:* `https://en.wikipedia.org/wiki/Accuracy_and_precision`, [Último acceso: mayo 2017].

[22] Auc. *En línea. Disponible:* `https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve`, [Último acceso: mayo 2017].

[23] Roc. *En línea. Disponible:* `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`, [Último acceso: mayo 2017].

[24] ¿qué es una base de datos relacional? *En línea. Disponible:* `https://aws.amazon.com/es/relational-database/`, [Último acceso: mayo 2017].

[25] Sql. *En línea. Disponible:* `https://es.wikipedia.org/wiki/SQL`, [Último acceso: mayo 2017].

[26] Mysql. *En línea. Disponible:* `https://www.mysql.com/`, [Último acceso: mayo 2017].

[27] Oracle 12c. *En línea. Disponible:* `https://www.oracle.com/es/corporate/features/database-12c/index.html`, [Último acceso: mayo 2017].

[28] Sqlite. *En línea. Disponible:* `https://www.sqlite.org/`, [Último acceso: mayo 2017].

[29] Postgresql. *En línea. Disponible:* `http://www.postgresql.org/`, [Último acceso: mayo 2017].

[30] Nosql databases. *En línea. Disponible:* `http://www.datastax.com/nosql-databases`, [Último acceso: mayo 2017].

[31] Mongodb. *En línea. Disponible:* `https://www.mongodb.com/`, [Último acceso: mayo 2017].

[32] Apache cassandra. *En línea. Disponible:* `http://cassandra.apache.org/`, [Último acceso: mayo 2017].

[33] Apache couchdb. *En línea. Disponible:* `http://couchdb.apache.org/`, [Último acceso: mayo 2017].

[34] Y. Freund and R. E. Schapire. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence, 14(5).*, pages 771–780, September, 1999.

[35] J. H. Friedman. Stochastic Gradient Boosting. March 26, 1999.

[36] Kaggle: Your home for data science. *En línea. Disponible:* `https://www.kaggle.com/`, [Último acceso: mayo 2017].

[37] E.J. Carmona Suárez. Tutorial sobre Máquinas Vectores Soporte (SVM). 2014.

[38] Events in the tracking logs. *En línea. Disponible:* `http://edx.readthedocs.io/projects/devdata/en/latest/internal_data_formats/tracking_logs.html`, [Último acceso: mayo 2017].

[39] Los 10 mandamientos de la usabilidad. *En línea. Disponible:* `http://mundoux.com/jakob-nielsen-principios-heuristicos`, [Último acceso: mayo 2017].

A

Estructura de paquetes de datos de edX

En esta sección se exponen los distintos tipos de datos en crudo que conformarán un curso MOOC.

Se realizará esta explicación tomando como ejemplo un curso MOOC cualquiera. Se empezará explicando la estructura de ficheros que se ha creado para el curso así como los ficheros que están contenidos en cada directorio. Una vez hecho esto, se explicará cada tipo de fichero que se dispone y cómo se ha distribuido en la estructura de ficheros creada.

A.1. Estructura de ficheros de un curso

Un curso MOOC consta de 5 directorios:

- Directorio *certificates*: En este directorio se alojará el fichero CSV de certificados.
- Directorio *course*: En este directorio se alojarán dos tipos de fichero relacionados con información básica del curso: un fichero de estructura del curso y un fichero de analíticas de los usuarios del curso, el cual no se utilizará.
- Directorio *events*: En este directorio se guardarán los distintos ficheros JSON del curso.
- Directorio *profile*: En este directorio se guardará un CSV que contiene diferentes datos del perfil de cada usuario; no será utilizado por la aplicación.
- Directorio *social*: En este directorio se guardará un CSV que contiene información sobre los distintos mensajes escritos en el foro. Puesto que estos mensajes son redundantes y aparecen en los ficheros de eventos JSON, no será utilizado por la aplicación.

Por lo tanto solamente tendremos que explicar la estructura de los ficheros de certificados y de los ficheros de eventos, además de cómo se ha utilizado el fichero de estructura del curso.

A.2. Certificados

El fichero de certificados es un fichero CSV cuyo nombre tiene el siguiente formato: *CertificateXXXYYYZZZ.csv*, donde *XXX* es el nombre corto del curso, *YYY* es el identificador del curso y *ZZZ* es el identificador de la edición. Este fichero está formado por 13 columnas, de las cuales solamente nos interesan las siguientes:

- *user_id*: Identificador del usuario.
- *grade*: Nota final del usuario.
- *status*: Tipo de certificado obtenido. Este campo se ha usado para determinar si el usuario aprobaba el curso o no. Hasta la última edición de los cursos del año 2016, este campo podía tener solamente dos valores: *notpassing* (no aprobado) y *downloadable* (aprobado). Sin embargo, para la última edición se han añadido nuevos valores posibles, por lo que surgió la necesidad de hacer compatible la aplicación tanto para los cursos antiguos como para los nuevos. Los nuevos valores son: *audit_passing* (aprobado), *audit_notpassing* (no aprobado) y *unverified* (se han considerado aprobados, pues se ha llegado a la conclusión de que se trata de usuarios aprobados que solicitaron certificado pero no lo llegaron a pagar)

La figura A.1 muestra un pequeño ejemplo del contenido que se puede encontrar en este fichero. Se han ocultado datos no anonimizados.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	user_id	download_url	grade	course_id	key	distinction	status	verify_uuid	download_uuid	created_date	modified_date	error_reason
2	15906225	29298		0.0	course-v1:		0	notpassing			2016-05-03 19:58:21	2016-05-03 19:58:21	
3	15906692	31890		0.0	course-v1:		0	notpassing			2016-05-03 19:59:25	2016-05-03 19:59:25	
4	15906762	32249		0.0	course-v1:		0	notpassing			2016-05-03 19:59:35	2016-05-03 19:59:35	
5	15901082	35718		0.0	course-v1:		0	notpassing			2016-05-03 19:46:42	2016-05-03 19:46:42	
6	15901566	38275		0.0	course-v1:		0	notpassing			2016-05-03 19:47:48	2016-05-03 19:47:48	
7	15901741	39237		0.0	course-v1:		0	notpassing			2016-05-03 19:48:11	2016-05-03 19:48:11	
8	15902373	42820		0.0	course-v1:		0	notpassing			2016-05-03 19:49:38	2016-05-03 19:49:38	
9	15902746	45082		0.0	course-v1:		0	notpassing			2016-05-03 19:50:30	2016-05-03 19:50:30	
10	15903352	48272		0.0	course-v1:		0	notpassing			2016-05-03 19:51:52	2016-05-03 19:51:52	
11	15904091	51803		0.0	course-v1:		0	notpassing			2016-05-03 19:53:33	2016-05-03 19:53:33	
12	15905186	53160		0.0	course-v1:		0	notpassing			2016-05-03 19:56:01	2016-05-03 19:56:01	
13	15905501	58453		0.0	course-v1:		0	notpassing			2016-05-03 19:56:43	2016-05-03 19:56:43	
14	15905871	60380		0.22	course-v1:		0	notpassing			2016-05-03 19:57:33	2016-05-03 19:57:33	

Figura A.1: Ejemplo de fichero csv de certificados

A.3. Ficheros de eventos

Los ficheros de eventos son ficheros JSON que guardan todos los eventos generados un día concreto a partir de la interacción de todos los estudiantes matriculados en dicho curso. Estos ficheros de eventos tiene el siguiente formato: *ZZZanonymized_XXXYYY-org-edx-date-edx.txt*, donde *XXX*, *YYY* y *ZZZ* representan lo mismo que en el fichero de certificados, org representa la organización que ha creado el curso (uamx por ejemplo) y date representa la fecha a la que pertenece dicho fichero con el formato *yyyy-mm-dd*.

En principio se dispone de un conjunto de ficheros de eventos asociados a la edición de un curso, pero solamente nos interesan los ficheros de eventos asociados a los días en los que se ha impartido el curso. Para ello se puede consultar el fichero de estructura del curso, que es un fichero JSON con distintos campos que definen todas las propiedades del curso, entre las que se encuentra la fecha de inicio y la fecha de finalización. Luego basta quedarse con los eventos que estén en ese rango de fechas y comprobar que no falta ninguno.

Una vez explicado cómo se han seleccionado los ficheros de eventos que se utilizarán, explicaremos su contenido. Básicamente cada fichero está formado por un array de objetos JSON, cada uno de los cuales está escrito en una línea del fichero. En la figura A.2 se muestra un ejemplo de evento generado.


```
{
  "username":"9822301",
  "event_source":"browser",
  "name":"play_video",
  "accept_language":"es-ES",
  "time":"2016-03-03T06:32:11.367740+00:00",
  "agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64 AppleWebKit/537.36 (KHTML, like Gecko Chrome/42.0.2311.135 Safari/537.36 Edge/12.10240",
  "page":"https://courses.edx.org/courses/course-v1:[redacted]/courseware/409277c05b1c40a68a6befb0714ac4c2/0c62a60c4a73463d8b533ba82113f12f/",
  "host":"courses.edx.org",
  "session":"0a824b86a58fd46cc466826a65177efa",
  "referer":"https://courses.edx.org/courses/course-v1:[redacted]/409277c05b1c40a68a6befb0714ac4c2/0c62a60c4a73463d8b533ba82113f12f/",
  "context":{
    "user_id":9822301,
    "org_id":"UAMx",
    "course_id":"course-v1:[redacted]",
    "path":"/event"
  },
  "ip":"aeb91e0e9529b1a8b4a741c5f4c850f6",
  "event":"{\"code\": \"0wP4c1zPrS0\", \"id\": \"ef1b85d3e648474fb17c1ffabdc7d8c4\", \"currentTime\": 0.111625}",
  "event_type":"play_video"
}
```


Figura A.2: Ejemplo de evento json de vídeo

De entre toda la información que alberga este objeto JSON, a la hora de generar los indicadores solamente se han usado los siguientes campos:

El campo **context.user_id** contiene el identificador del usuario que ha generado el evento o para el cual el sistema ha generado el evento. En principio el identificador de usuario se obtenía del campo *user_id*, pero se observó que para determinados cursos ese campo estaba vacío.

El campo **time** contiene la fecha y hora en formato UTC en la que se ha registrado el evento. Tiene el siguiente formato: *"year-month-dayTh-m-s.ms"*. Este campo se ha utilizado para la generación de indicadores relacionados con el tiempo y las sesiones.

El campo **event_type** contiene un identificador que describe el tipo de evento. Este campo será de gran importancia para la generación de indicadores, pues nos permite clasificar el evento.

En general existen dos tipos generales de eventos en función de su generación. Por una parte están los eventos que llamaremos indefinidos, que se generan para indicar una petición GET o POST a la plataforma; por otra parte están los eventos que llamaremos definidos, generados como respuesta del sistema a estas peticiones, o generados por el sistema en sí. En general nos interesan mucho más estos últimos, aunque usaremos todos a la hora de generar indicadores que cuentan eventos.

En el caso de que el evento sea definido, el campo event_time proporcionará una definición concreta del evento. Esta información puede consultarse en detalle en [38], donde define la funcionalidad concreta del evento y en qué campos se encuentra la información asociada al evento.

La generación de los indicadores desarrollados en el proyecto se han basado en la clasificación de este tipo de eventos que realiza edX, que clasifica toda la lista de tipos de eventos por su funcionalidad. Solamente hemos tenido en cuenta las siguientes clases de eventos:

- Navigation Events: Son eventos relacionados con la navegación por el propio curso. La lista de tipos de eventos asociados a esta clase son los siguientes:
  - edx.ui.lms.link_clicked
  - edx.ui.lms.outline.selected
  - edx.ui.lms.sequence.next_selected
  - edx.ui.lms.sequence.previous_selected
  - edx.ui.lms.sequence.tab_selected
  - page_close
- Video Interaction Events: Son eventos relacionados con la interacción en vídeos. La lista de tipos de eventos asociados a esta clase son:
  - hide_transcript
  - edx.video.transcript.hidden
  - edx.video.closed_captions.hidden
  - edx.video.closed_captions.shown
  - load_video
  - edx.video.loaded
  - pause_video
  - edx.video.paused
  - play_video
  - edx.video.played
  - seek_video
  - edx.video.position.changed
  - show_transcript
  - edx.video.transcript.shown
  - speed_change_video
  - stop_video
  - edx.video.stopped
  - video_hide_cc_menu
  - edx.video.language_menu.hidden

  - video_show_cc_menu
  - edx.video.language_menu.shown

- Forum Interaction Events: Son eventos relacionados con la interacción en foros. La lista de eventos asociados a esta clase son:
  - edx.forum.comment.created
  - edx.forum.response.created
  - edx.forum.response.voted
  - edx.forum.searched
  - edx.forum.thread.created
  - edx.forum.thread.voted
- Problem Interaction Events: Son eventos relacionados con la interacción en los problemas. La lista de eventos asociados a esta clase son:
  - edx.problem.hint.demandhint_displayed
  - edx.problem.hint.feedback_displayed
  - problem_check
  - problem_check_fail
  - problem_graded
  - problem_rescore
  - problem_rescore_fail
  - problem_reset
  - problem_save
  - problem_show
  - reset_problem
  - reset_problem_fail
  - save_problem_fail
  - save_problem_success
  - showanswer

B

Manual de usuario de la herramienta

En este anexo se proporciona un manual de usuario de acuerdo con los distintos casos de uso detallados en 3.

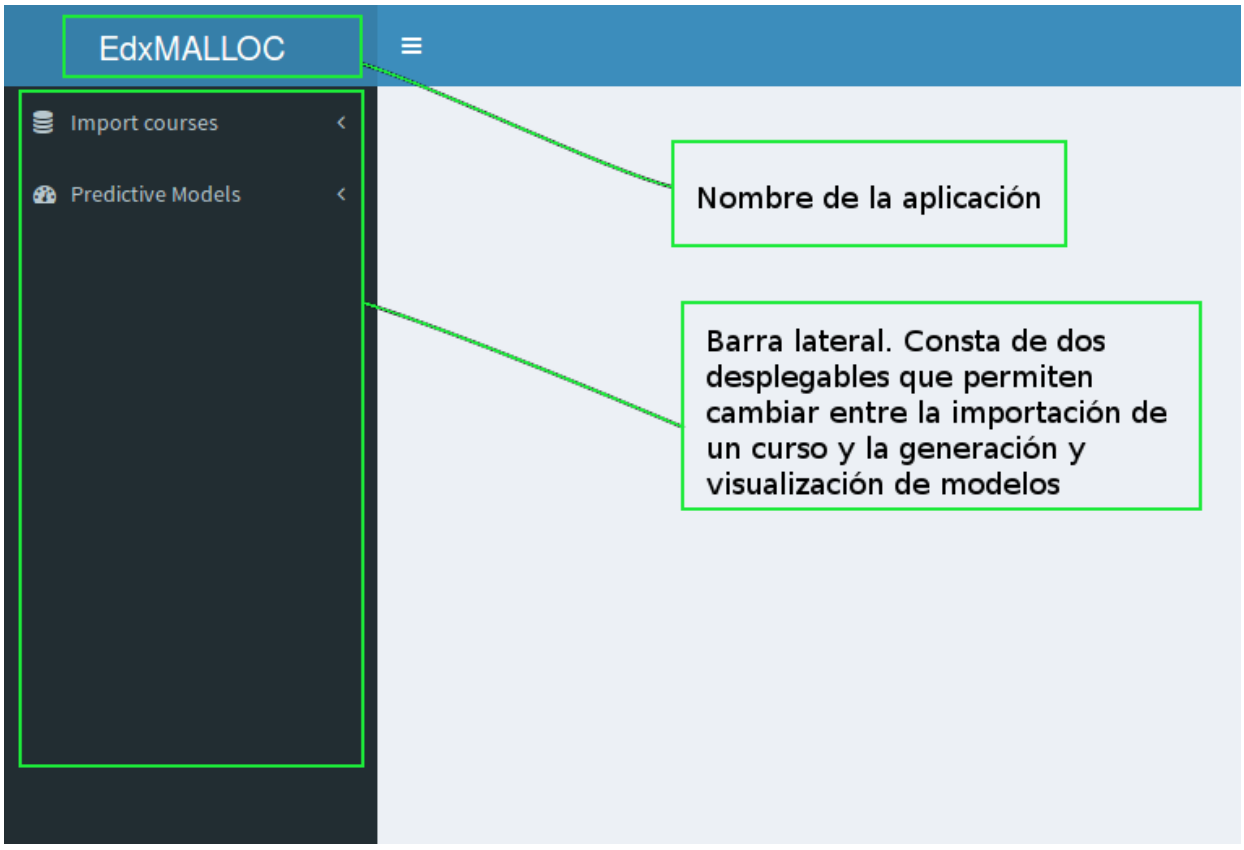

Figura B.1: Pantalla principal

B.1. RF-01: Importar datos de una edición de un curso

Figura B.2: Pantalla principal para la importación de datos

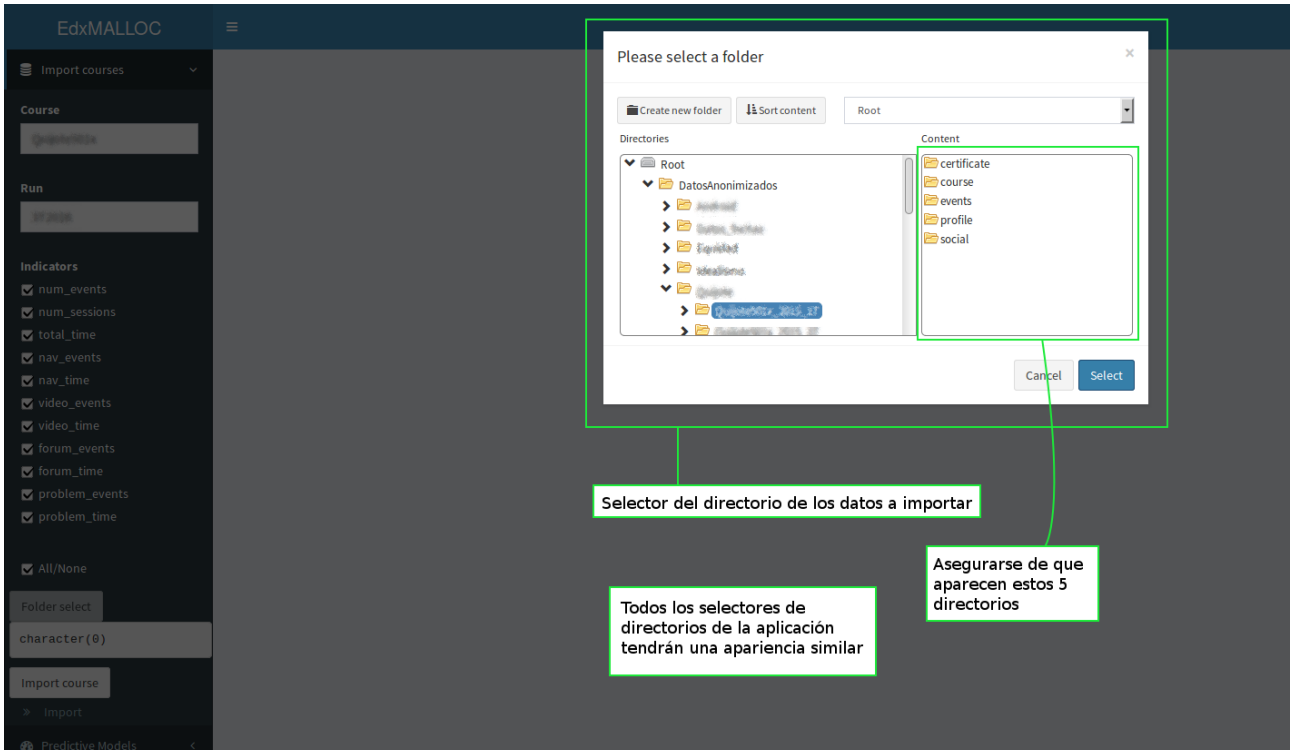

Figura B.3: Ventana de selección de la carpeta de datos de un curso

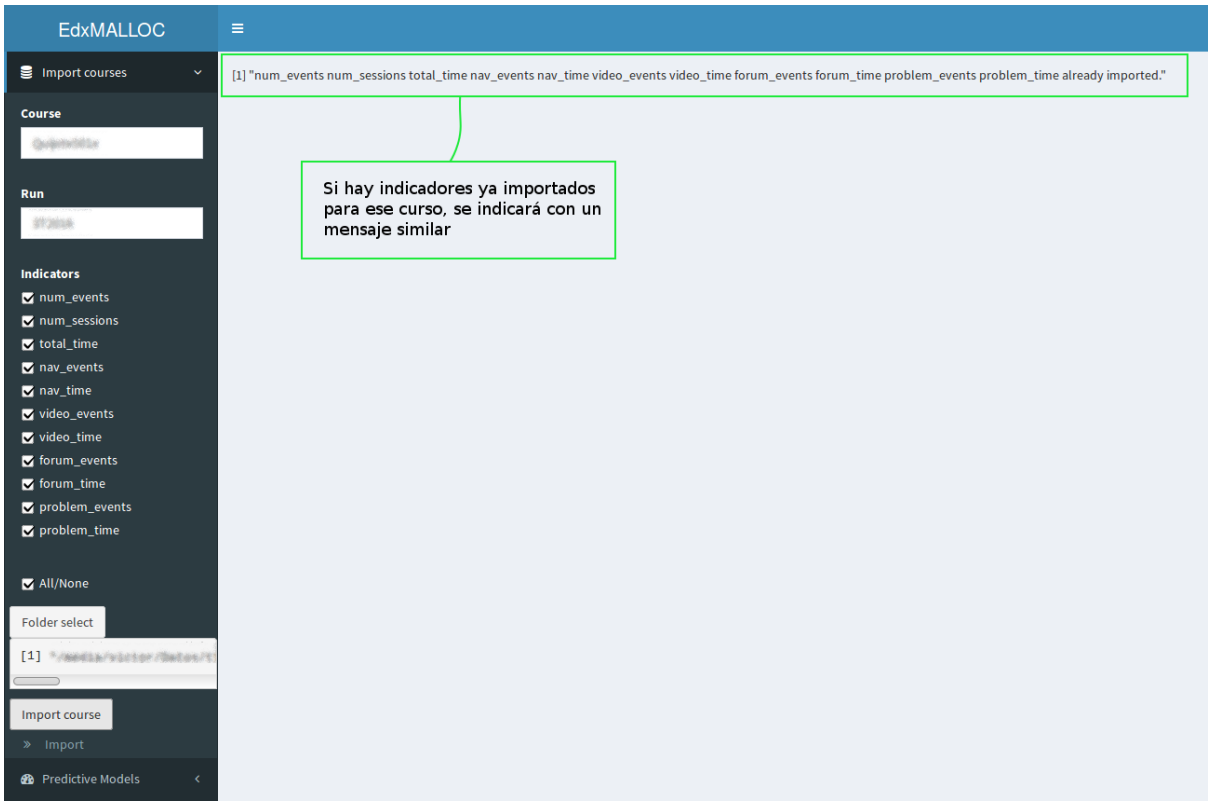

Figura B.4: Mensaje de error para indicadores de un curso ya importados

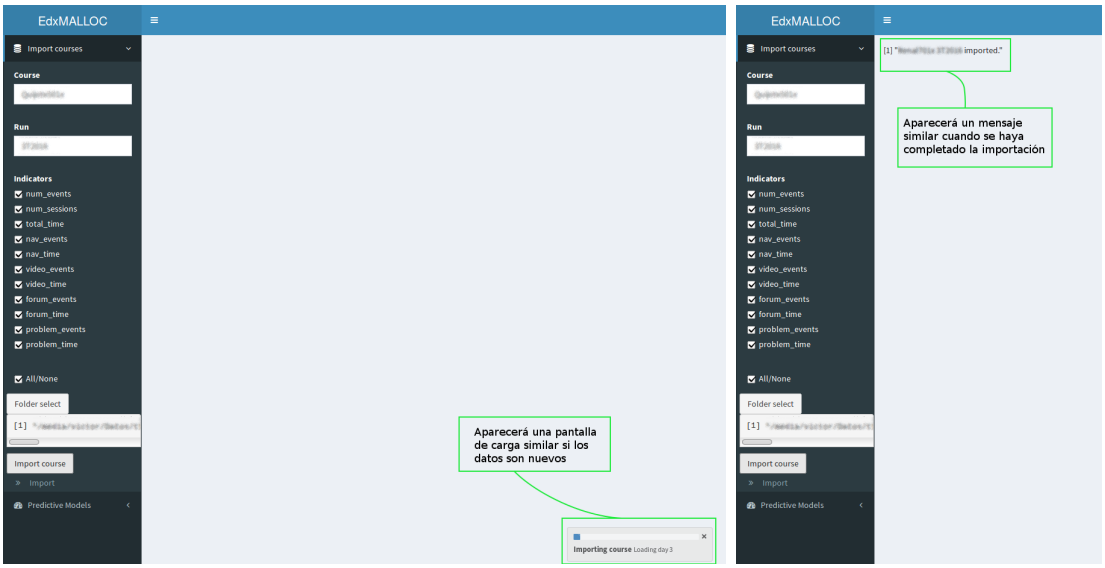

Figura B.5: Pantalla de carga de un curso y mensaje de importación con éxito

B.2. RF-02: Generar y guardar un modelo predictivo para una edición de un curso

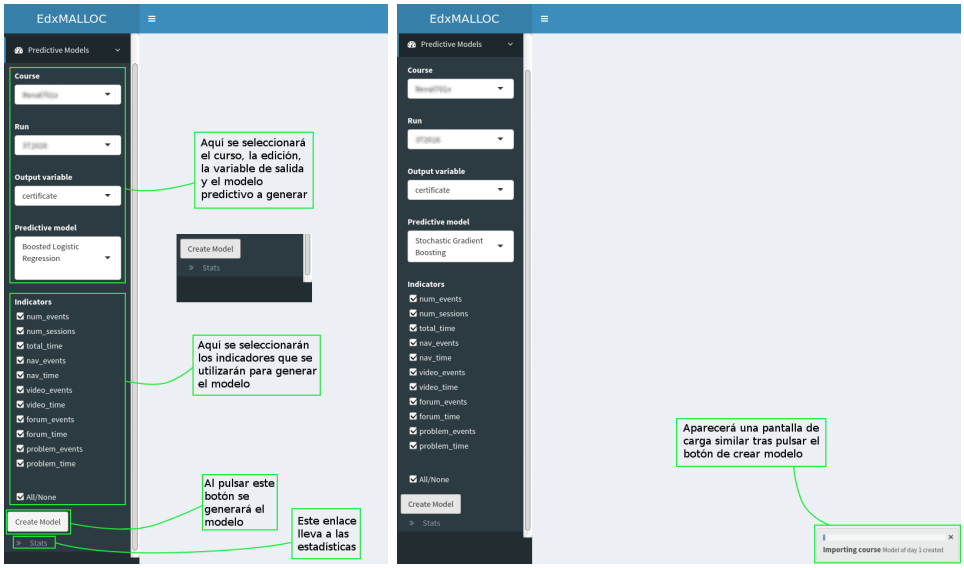

Figura B.6: Pantalla principal para la generación de modelos y pantalla de progreso

B.3. RF-03-04-05: Visualización y exportación de estadísticas y selección del mejor modelo

Figura B.7: Visualización de estadísticas temporales, AUC y selección del mejor modelo

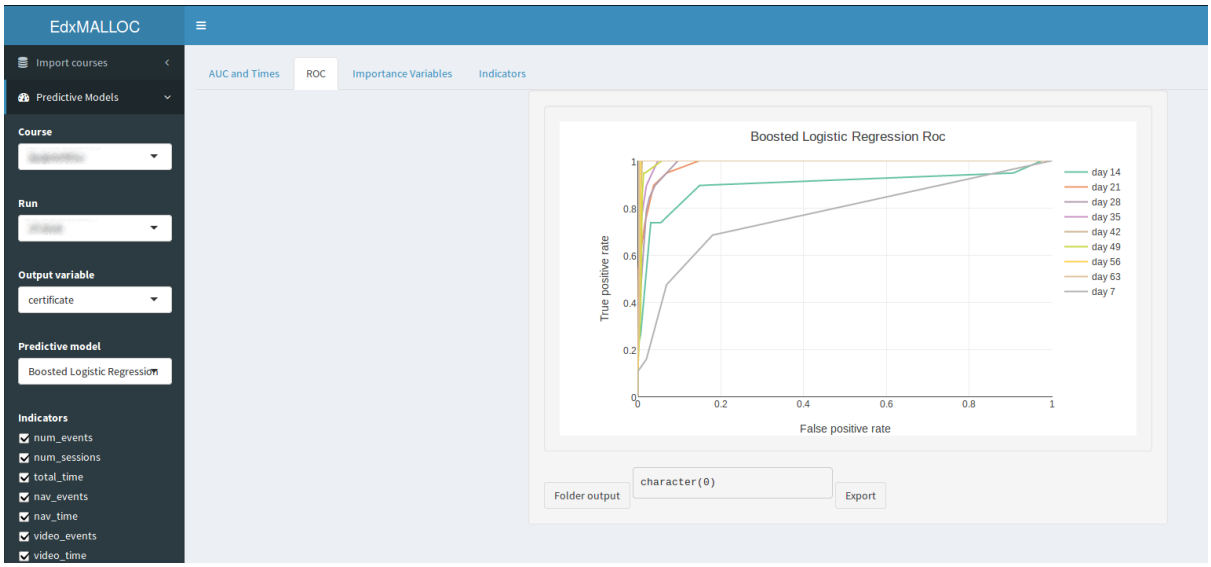

Figura B.8: Visualización de curvas ROC

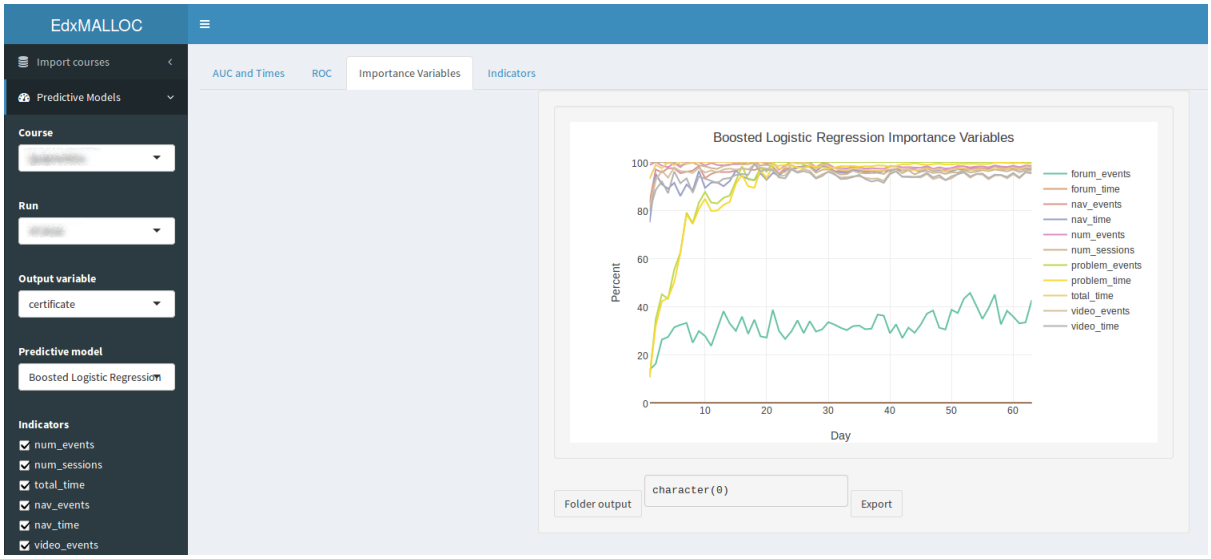

Figura B.9: Visualización de la importancia de las variables

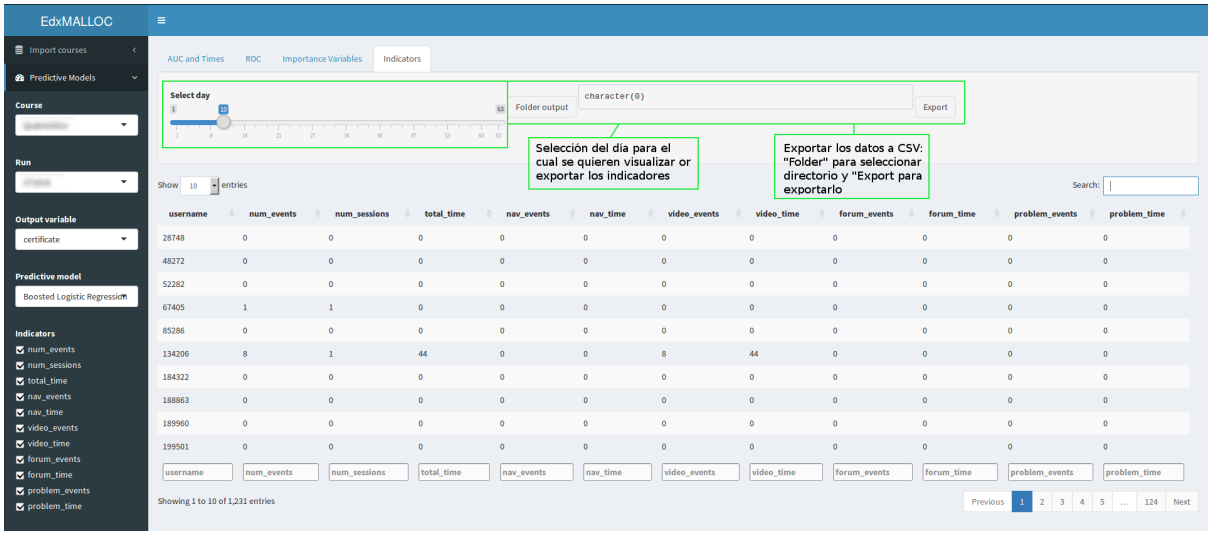

Figura B.10: Visualización de la pantalla de indicadores

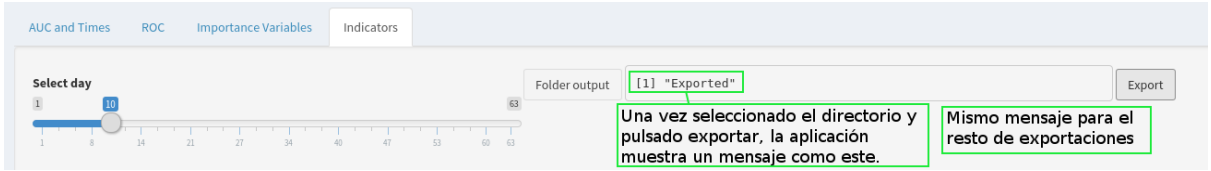

Figura B.11: Visualización de la respuesta una vez exportados los datos

C

Encuesta de usabilidad

En este anexo se presenta la encuesta de usabilidad creada para probar la usabilidad de la aplicación diseñada. Es una encuesta básica utilizando las 10 preguntas de usabilidad de Nielsen [39] y añadiendo 3 preguntas abiertas que han parecido interesantes. Se ha realizado usando la herramienta de formularios de Google, que permite fácilmente sacar estadísticas de las respuestas.

En la figura C.1 se muestra la encuesta creada.

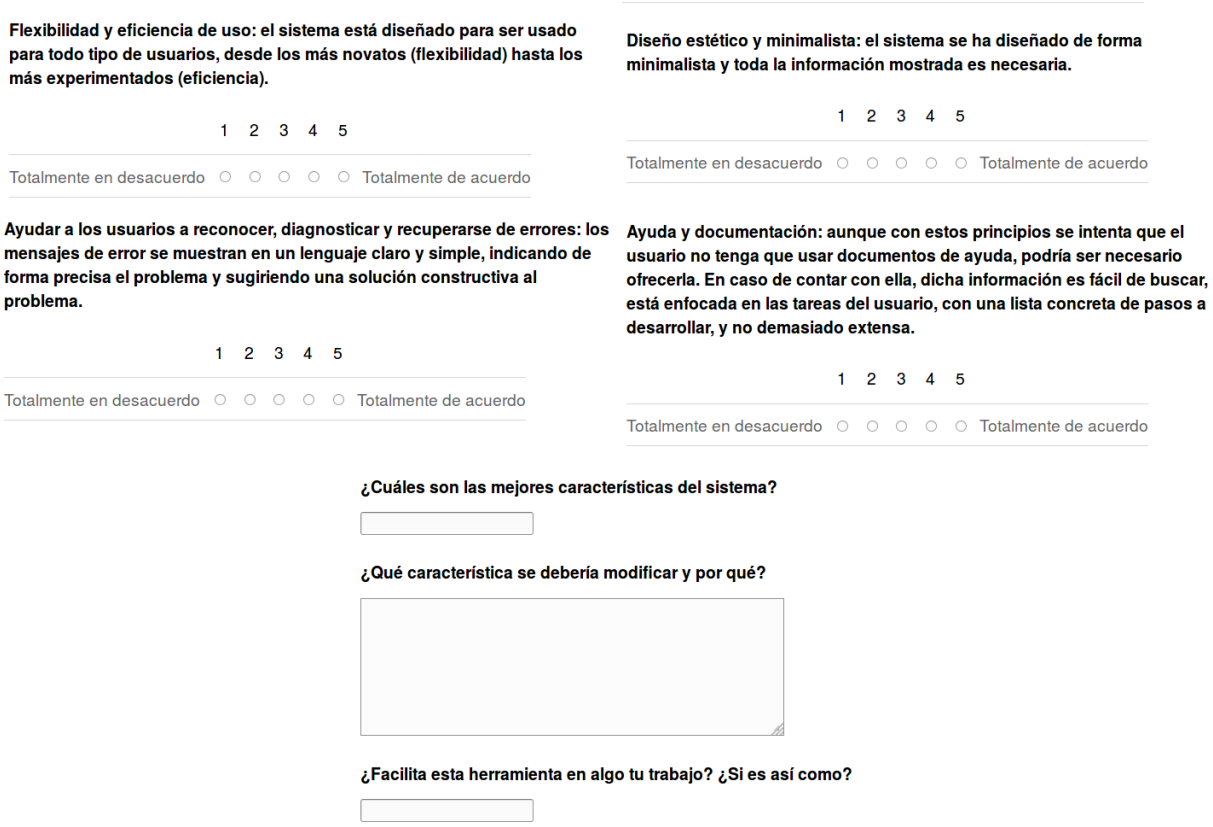

Figura C.1: Encuesta de usabilidad

A continuación se muestran los resultados actuales de la encuesta.

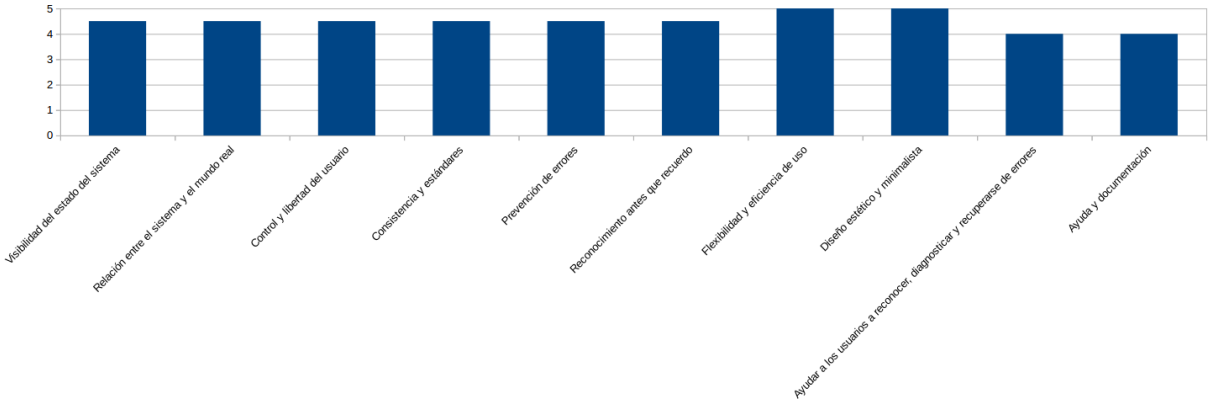

Figura C.2: Encuesta de usabilidad