Ingenieria Informática

3º curso

Teacheck

Autores

Unai Agirre, Asier de La Natividad, Jon Fernandez y Lucas Sousa

09-06-2019

Abstract

“Teacheck surgió de la necesidad de un sistema capaz de detectar el deterioro en el rendimiento de un alumno en una universidad de forma precisa y eficaz. El siguiente informe contempla el análisis, definición y visualización de la información necesaria para la correcta aplicación de este concepto. La solución planteada agrupa la enseñanza automática requerida para el análisis, el servicio web para su visualización y la estructura de datos que será clave para mantener el flujo de datos para que el análisis estadístico de los resultados sea lo más preciso y eficaz posible. A continuación, se recoge tanto el informe de los aspectos y herramientas técnicas utilizadas para el desarrollo de la ampliación, como la valoración de las competencias de la aplicación desarrolladas durante esas tareas.”

Contents

List of Figures

1. CHAPTER

Introducción

1.1 Teacheck

Teacheck es una aplicación para realizar el seguimiento del alumnado de una universidad. Esta aplicación contará con un sistema para analizar el rendimiento y detectar a tiempo un deterioro del mismo. En base a esto, nuestro objetivo es que gracias a los servicios proporcionados se consiga revertir la situación y mejorar tanto el rendimiento como motivación del alumno en concreto. También queremos monitorizar todo el proceso de seguimiento para así poder aligerar la carga de trabajo del profesorado. Como último objetivo tenemos el detectar no solo que el alumno necesita un toque de atención, si no también saber cual es la raíz del problema y a su vez podremos detectar problemas a nivel de clase, esto es, si en una misma clase vemos que se dan los mismos problemas en demasiados casos, se alertará de ello para que el cliente esté al tanto y pueda buscar una solución.

La solución no se centrará en mejorar cómo se imparten las clases, o en hacerle entender mejor al alumno la materia, sino en un proceso el cual analiza y monitoriza los resultados individuales de cada alumno con el fin de encontrar los posibles problemas a tiempo. Esto no quita que nuestros análisis no puedan detectar un posible problema a nivel de clase o de entendimiento del alumno, pero la solución de ese problema no será nuestra responsabilidad. No pretendemos cambiar la forma de funcionar de una institución, sino mejorarla.

La aplicación web contará con un sistema distribuido que será el responsable de proveer

servicios básicos a la aplicación o puede que algún servicio adicional acordado con el cliente, se implemente con el sistema. Dicho sistema estará dividido en varios microservicios que ofrecen recursos esenciales a la aplicación principal. Servicios como, Base de Datos, el proveedor de datos de la Universidad, el sistema de IA Machine Learning y como dicho antes si es de interés del cliente, se podrá amoldar a algún sistema que existe y que ya provee todos los datos de los alumnos.

Teacheck, es un sistema de monitoramento que ofrecerá periódicamente análisis sobre la situación de cada alumno registrado en la universidad. Contará con el sistema de Machine Learning (Inteligencia Artificial) para efectuar los análisis. Estos escaneos de información son específicamente predicciones en base a un modelo que ya dispone el sistema con la información inicial proporcionada por el cliente.

Cada análisis traerá consigo resultados sobre cada alumno en respecto a su rendimiento en el curso correspondiente. Contaremos tanto con los datos recogidos por el profesor como con los que el alumno pueda proporcionar. Por parte del profesorado recogeremos las notas, asistencia y si el alumno hace entrega o no de los trabajos que se mandan. También tendrá como opcionales la atención, motivación y la nota del alumno en los entregables. Luego, el alumno tendrá como tarea opcional hacer un feedback rápido en el cual podrá ofrecer a la aplicación su atención en clase, motivación, asistencia y tiempo dedicado a las asignaturas fuera del horario de clase. Es a tener en cuenta que si los datos opcionales se entregan, se conseguirá un análisis mucho más preciso y eficaz, por eso son importantes unas cuantas claves que más tarde se mencionan en el apartado de actividades y recursos clave.

Teacheck tendrá un sistema de alarmas que funcionará según el resultado de los análisis. Estas alarmas servirán para avisar a los profesores, vía email y aplicación, sobre un estado crítico o sobre la necesidad de darle un toque a un alumno. Entonces, si después de un análisis la aplicación detecta una de las distintas alarmas que tenemos diseñadas, las cuales se dividen en dos grupos, avisará al profesor o responsable correspondiente. Tal y como se acaba de mencionar las alarmas están divididas en dos grandes grupos, uno para alertar de un alumno y otro para alertar de un posible problema a nivel clase. Por parte de las alertas del alumno, se recibirán cuando se note un bajo rendimiento tanto en una asignatura como en el curso en general. Luego se analizará el rendimiento general de una clase por lo tanto, en caso de que baje demasiado saltará una alerta, a la vez que con el nivel de satisfacción, que en caso de que el nivel sea bajo también se podrá detectar. Por último la aplicación tendrá un calendario por curso donde los profesores introducirán sus exámenes correspondientes para que así la aplicación pueda realizar un escaneo una

semana antes del mismo y así el profesor pueda ver si el alumno está preparado o no.

La aplicación dispone de dos roles distintos:

- **Alumnos:** Podrá ver su seguimiento a diario y además, proveer datos al sistema de la aplicación ya que se le pedirá rellenar una encuesta semanalmente. Dicha encuesta preguntará detalles sobre el estado actual del alumno y su nivel de satisfacción en relación a las actividades lectivas.
- **Profesores:** Podrá monitorizar sus alumnos y cuando sea el caso recibirá alarmas de alumnos que se han detectado con bajo rendimiento o que necesiten atención.

1.2 Mercado

La aplicación está dirigida a universidades que busquen hacer un seguimiento detallado de su alumnado. El usuario final del sistema, por tanto, serán tanto los diferentes grupos de profesores o coordinadores como los alumnos que se encuentren en la institución. El ámbito geográfico que pretendemos abarcar es el de nivel nacional. Tras analizar diferentes aplicaciones hemos visto que se dividen en distintas categorías:

- Aplicaciones con IA que ayudan a los alumnos a aprender de una manera más eficiente y efectiva.
  - **Easy learning:** Kidaptive es una plataforma de enseñanza adaptativa que impulsa una variedad de dominios de aprendizaje que incluyen dos aplicaciones creadas por Kidaptive para el aprendizaje temprano. Osmo es un juego interactivo que combina aprendizaje online y experiencial.
- Aplicaciones que ayudan a los profesores en la enseñanza con la ayuda de la IA.
  - **Contenido:** Los proveedores de contenido premium utilizan cada vez más el aprendizaje automático para ofrecer la siguiente mejor lección. Por ejemplo, startups como Content Technologies Inc. hacen uso de machine learning para automatizar su producción y automatización de procesos de negocio, diseño instruccional y soluciones de contenido y el proceso de enseñanza.
- Aplicaciones sin IA que automatizan las actividades de monitoreo del profesor en respecto al alumnado.[**?**]

  - **Additio:** Se trata de una herramienta versátil con muchas funcionalidades al alrededor del mundo educativo, entre ellas la capacidad de llevar un registro de notas de los estudiantes de forma muy visual, intuitiva y práctica.
  - **TeacherKit:** Permite crear diferentes clases, cada una con sus alumnos y un sinfín de opciones para cada una de ellas. TeacherKit ayuda a llevar un registro de notas y también de asistencias y de comportamiento, con la posibilidad de exportar todos los datos para gestionarlos por su cuenta.

- Aplicaciones con IA que monitorean el rendimiento de los alumnos y sacan alarmas según los diferentes objetivos:

  - Aplicaciones que tienen como objetivo prevenir el abandono de alumnos en respecto a su carrera.
    * **Universidad de Derby:** donde se implementó un sistema de monitoreo de la deserción estudiantil que utiliza los datos para predecir qué estudiantes tienen riesgo de dejar sus estudios, permitiéndole a la institución intervenir antes de que ello suceda.[**?**]
  - Aplicaciones que previenen el deterioro del rendimiento del alumnado con el fin de revertir la mala situación para obtener mejores resultados.
    * **Universidad Internacional de la Rioja:** Un equipo de expertos de la Universidad Internacional de La Rioja trabaja en un proyecto piloto para, gracias al uso y aplicación de la Inteligencia Artificial (IA), poder medir ,mediante algoritmos, dicho rendimiento. Se analiza el comportamiento del alumno en la plataforma, su participación en los foros, su interacción con el material de estudio, las calificaciones intermedias obtenidas en la evaluación continua... Al poder compararse con los históricos de estudiantes anteriores, se observa si existe un patrón.[**?**]

Teacheck se situa en la categoría de aplicaciones que intentan prevenir el deterioro del rendimiento académico de los alumnos. Dentro de esta categoría ya existe una institución que realiza este tipo de actividad. Pero Teacheck ofrece algo más que solamente el análisis de los datos proporcionados por el profesor, también ofrece la posibilidad de que un alumno pueda proporcionar datos los cuales permitirá a la aplicación ser más precisa y eficaz. Ya que de esta manera, los datos introducidos por los profesores nos dirán en qué está fallando el alumno y los datos proporcionados por el alumno, cómo solucionarlo.

1.3 Propuesta de Valor

Teacheck es una aplicación con el objetivo de facilitar un seguimiento en beneficio tanto del alumnado principalmente como del profesor. Con esto pretendemos resolver el problema que actualmente hay con el alto porcentaje de repetidores y suspensos. Creemos que la mayoría de estos casos suceden por falta de responsabilidad y desconocimiento de cuando el alumno va mal o su rendimiento no es el adecuado, y gran parte de estos casos son evitables. Y para dar solución a esto el primer paso es el interés del profesorado en intentar revertir esta situación y tratar de alertar al alumno de su situación, que aunque él ya sea consciente de ello en la mayoría de los casos, el hecho de recibir un aviso o consejo de forma adecuada, esto es, teniendo en cuenta cual es la manera correcta de decir y plantear los problemas, puede hacerle cambiar. Por lo tanto, para conseguir esto es necesario el seguimiento del alumno en cuestión y poder tanto ver cómo tener presente sus notas, asistencia, motivación y sus entregables entre otros. Pero aquí se nos plantea otro problema, y es que hacer el seguimiento de un alumno es fácil, pero no es lo mismo con diez alumnos, o veinte, o treinta. Entonces se complican las cosas ya que el profesor o tutor en cuestión no podrá cumplir con el seguimiento de todos y los avisos no podrán llegar a tiempo.

Pero para todo existe una solución y en este caso ofrecemos Teacheck, tal y como hemos comentado al principio, una aplicación para realizar el seguimiento del alumnado, de forma automática y precisa. El objetivo es reducir en gran parte los repetidores y suspensos para así aumentar el rendimiento de los alumnos. También vamos a poder analizar diferentes problemas que puedan surgir a nivel de clase gracias a los diferentes datos que almacenaremos. Con todo esto conseguiremos aumentar el rendimiento de las clases y en consecuencia el de la universidad en general, afectando positivamente tanto en su prestigio como en su valor como institución lo que atraerá a nuevos alumnos y empresas.

1.4 Actividades Clave

Tras un análisis detallado de la institución en la que se va a desarrollar el sistema creemos que los puntos clave a la hora de implementarlo y que aportarán valor, son los siguientes:

- **Atributos a analizar:** Se debe definir y concretar los atributos que se tendrán en

cuenta en el machine learning, ya que estos serán los que en un futuro se valorarán y relacionarán entre ellos para sacar conclusiones tanto de las clases como de los alumnos.

- **Alarmas:** Las alarmas que los atributos mencionados en el punto anterior podrán llegar a generar deben ser claras y concisas, detectando así la raíz del problema a tiempo y aportando un punto de inicio a la hora de solventar el problema.

- **Informar y comunicar:** Creemos que lo primero es informar correctamente al alumno de lo que esta aplicación es y lo que le puede aportar. No es algo creado para controlarlo, si no algo que lo beneficiará si lo usa. Apenas le pide tiempo, solo unos pocos minutos a la semana y es importante que el alumno entienda esto para evitar malentendidos y descontentos. La aplicación seguirá funcionando sin sus aportaciones pero son esenciales para que este funcione al 100%. Además de esto, también es importante una vez salte un aviso, comunicarle lo ocurrido al alumno de forma correcta, ya que si no, no conseguiremos revertir la situación tal y como queremos que suceda.

2. CHAPTER

Análisis del Sistema

2.1 Definición de la empresa y del servicio

Teacheck ofrece a estos centros una aplicación web con la cual mediante sus servicios es capaz de realizar un seguimiento de sus alumnos, gracias a tecnología innovadora como la inteligencia artificial. Es una empresa destinada a instituciones o universidades tanto públicas como concertadas que busquen realizar una monitorización automática y precisa de su alumnado. Ofrece un servicio con el objetivo de facilitar un seguimiento en beneficio tanto del alumnado principalmente como del profesor. Con esto pretendemos resolver el problema que actualmente existe con el alto porcentaje de repetidores y suspensos. Creemos que la mayoría de estos casos suceden por falta de responsabilidad y desconocimiento de cuando el alumno va mal o su rendimiento no es el adecuado, y gran parte de estos casos son evitables. Y para dar solución a este problema, el primer paso es el interés del profesorado en intentar revertir esta situación y tratar de alertar al alumno de su situación, que, aunque él ya sea consciente de ello, en la mayoría de los casos, el hecho de recibir un aviso o consejo de forma adecuada, esto es, teniendo en cuenta cual es la manera correcta de decir y plantear los problemas, puede hacerle cambiar de actitud. Por lo tanto, para conseguir esto es necesario el seguimiento del alumno en cuestión y tener presentes sus notas, asistencia, motivación y sus entregables entre otros. Pero aquí se nos plantea otro problema, y es que hacer el seguimiento de un alumno es fácil, pero no es lo mismo con diez, veinte o treinta de ellos. Entonces se complican las cosas ya que el profesor o tutor en cuestión no podrá cumplir con el seguimiento de todos y los avisos

no podrán llegar a tiempo.

2.1.1 Organigrama de la empresa

Como podemos ver en el organigrama de abajo, la empresa está formada por 4 departamentos o unidades que se basan en:

- El producto, esto es,el desarrollo de toda la aplicación en general. Podríamos dividir este departamento en diferentes secciones o roles como podrían ser: El análisis y diseño, el desarrollo del software y el análisis estadístico y la inteligencia artifical del producto.
- Monitorizar y gestionar el capital de la empresa así como las inversiones.
- La gestión los empleados y lo relacionado con nuevas incorporaciones al equipo y su bienestar.
- Por último ,contactar nuevas universidades y promoción de la aplicación.

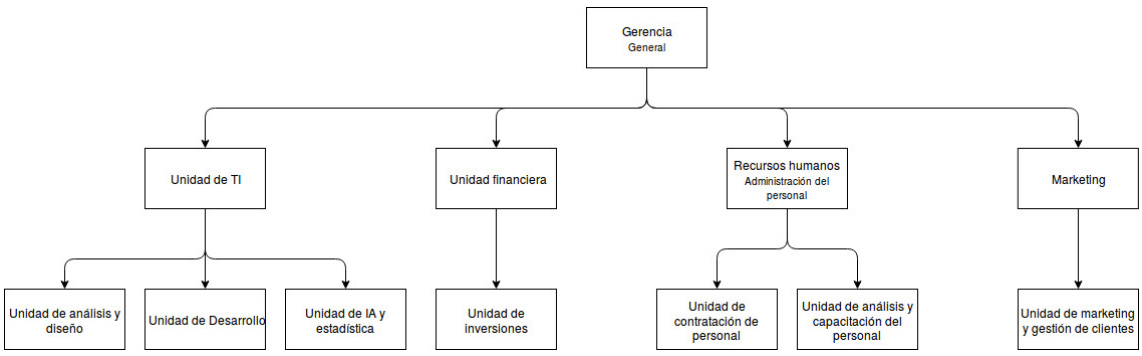


Figure 2.1: Organigrama de la empresa

2.1.2 Ventajas

Como hemos mencionado anteriormente, gracias al seguimiento personalizado que ofrece Teacheck, una universidad podrá organizar de una forma fácil y sencilla toda información relacionada con el rendimiento de un alumno o clase en general.Así el personal docente

y los alumnos tendrán una forma de detectar el motivo de una posible degradación en el ritmo de trabajo de un alumno o profesor. Además, como ha sido mencionado anteriormente, gracias a este sistema, se tendrán en cuenta toda la información de todos los alumnos de una clase para saber si una asignatura en concreto es implantada de forma correcta.

2.1.3 Desventajas

La participación del alumnado no es necesaria para el correcto funcionamiento del sistema, pero su eficiencia y exactitud incrementa exponencialmente si dispone de la información que un alumno puede llegar a ingresar. Es por esto que, aún siendo opcional, los alumnos se puedan sentir presionados o en cierta forma controlados por la universidad a la hora de tener que ingresar datos como puedan ser las horas de estudio fuera de clase, la asistencia o la motivación en el aula. El personal docente del centro deberá recoger tanto las notas como la asistencia diaria de sus alumnos y calificar cada clase que imparta, ya que después, las observaciones hechas durante esa clase, serán claves para saber el estado de sus alumnos, aumentando así las responsabilidades de un profesor con una asignatura.

2.1.4 Elementos diferenciadores

Teacheck se situa en la categoría de aplicaciones que intentan prevenir un deterioro en el rendimiento académico de los alumnos. Dentro de esta categoría ya existe una institución que realiza este tipo de actividad. Pero Teacheck ofrece algo más que solamente el análisis de los datos proporcionados por el profesor, también ofrece la posibilidad de que un alumno pueda proporcionar datos los cuales permitirá a la aplicación ser más precisa y eficaz. Ya que de esta manera, los datos introducidos por los profesores nos dirán en qué está fallando el alumno y los datos proporcionados por el alumno, cómo solucionarlo.

2.1.5 Gestión de activos

Teacheck es una pequeña cooperativa compuesta por 4 trabajadores que acaba de entrar en el mundo laboral. Las unidades de las que dispone la empresa son las siguientes: Unidad de TI, Unidad financiera, Recursos humanos y Marketing. Al ser una empresa de 4 trabajadores, cada uno forma parte de más de una unidad. Todos ellos son parte de la unidad de TI ya que son desarrolladores, pero entre las otras 3 unidades, se reparten los 4 técnicos.

Uno de ellos, trabaja en la Unidad Financiera, dos de ellos trabajan en Recursos Humanos y el último en Marketing.

	Unidad de TI	Unidad Financiera	Recursos Humanos	Marketing
Asier De la Natividad	X	X		
Jon Fernandez	X			X
Lucas Sousa	X		X	
Unai Agirre	X		X	

Figure 2.2: Departamentos y sus técnicos

Al ser una pequeña empresa y bastante nueva no dispone de muchos activos. La empresa dispone de 4 ordenadores portátiles, uno para cada empleado. Los empleados utilizarán los ordenadores tanto para el desarrollo de la aplicación Teacheck como para los demás trabajos como finanzas, marketing o recursos humanos. Al ser nueva en el mundo laboral y no tener financiación, la cooperativa no dispone de servidores propios.

2.1.6 Service desk

La función service desk es la encargada de registrar todas las incidencias y solicitudes de servicio de nuestra aplicación. Es el punto único de contacto entre los usuarios y los proveedores de servicio. Este servicio no solo será proporcionado a los usuarios de la aplicación, también será utilizado por los técnicos de la empresa.

Este servicio se ofrece mediante diferentes canales como:

- Teléfono: Los usuarios de la aplicación y los técnicos de la empresa podrán utilizar el teléfono como forma de comunicación para recibir información o dar avisos de problemas o incidencias.
- Correo electrónico: Podrán enviar correos electrónicos con fin de recibir información o avisos de incidencias.
- Redes sociales: Solo se utilizará para recibir información.

2.1.7 Gestión de incidencias

Teacheck, es una aplicación que será utilizada por muchas personas dependiendo de la cantidad de alumnos de dicha universidad. La aplicación estará en constante mantenimiento pero eso no asegura que puedan surgir nuevas incidencias.

Para ello, definiremos el proceso de gestión de incidencias, que es el proceso responsable de la gestión del ciclo de vida de las incidencias. Esto asegura que se restablezca la operación normal de servicio lo antes posible y se minimice el impacto negativo al negocio para que se mantenga el nivel de calidad del servicio acordado.

Proceso de gestión de incidencias

- **Identificar incidencia:** La primera actividad de este proceso será identificar la incidencia. Es una de las actividades claves ya que es muy importante saber de dónde viene esa incidencia para después poder arreglarla. Aquí se debatirá si de verdad es una incidencia y hay que tratarla o no.
- **Registrar incidencia:** El segundo paso será registrar la incidencia. Esto nos ayudará en un futuro a resolver de manera más fácil y rápida una nueva incidencia ya que es posible que esté registrada y solo deberemos seguir los pasos que se siguieron la anterior vez.
- **Clasificar y priorizar incidencia:** Una vez registrada pasaremos a clasificar y priorizar la incidencia. Estas, deberemos clasificarlas por categorías o niveles y priorizar esas categorías para ver qué incidencias son las que corren más prisa solucionar.
- **Asignación de incidencia:** Teniendo en cuenta la disponibilidad de los técnicos, se asignará la incidencia al experto en ese tipo de incidencias.
- **Investigar y diagnosticar incidencia:** Una vez asignada la incidencia, el técnico investigará acerca de la incidencia para saber cuál es el problema y que se debe hacer para solucionarla.
- **Resolver incidencia:** Solo queda solucionar la incidencia una vez que sabemos cual es el problema y como se soluciona.

- **Registrar solución y cerrar incidencia:** Por último, registraremos de manera redactada un informe con los pasos seguidos para solucionar el problema y cerraremos la incidencia.

2.2 Arquitectura del sistema

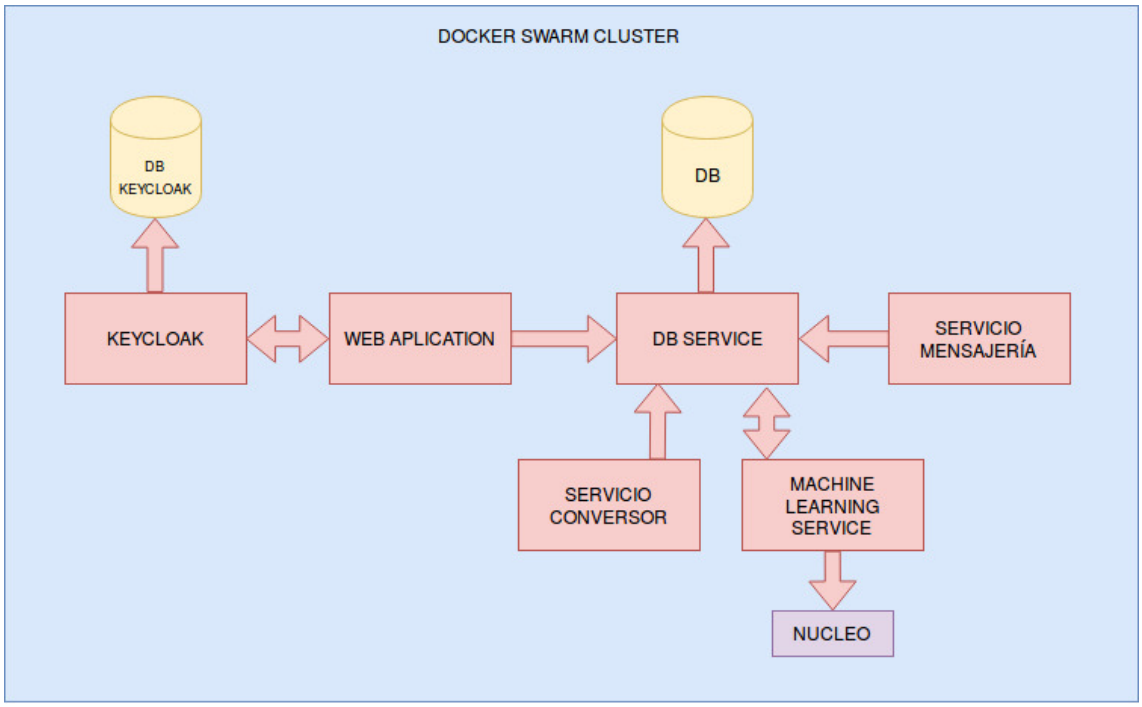


Figure 2.3: Arquitectura del Sistema

- **Web Application:** La aplicación web tiene el objetivo de visualizar los resultados proporcionados por el sistema de análisis de datos y monitorear los diferentes alumnos registrados en el sistema de Teacheck.
- **Keycloak:** Keycloak es un servicio que gestiona el acceso y identidad de entidades. Este servicio acoplado con la aplicación web para proveer autenticación y autorización a recursos.
- **Servicio Machine Learning:** El servicio de análisis de datos y predicción. Este sistema tiene el objetivo de analizar todos los datos proporcionados por la universidad

cliente en respecto a los alumnos. Contiene un núcleo que se trata de un servicio más pequeño que hace las predicciones semanalmente.

- **Servicio Conversor:** Servicio de conversión de formatos. El sistema no obliga al cliente adaptarse al formato de intercambio de información que dispone Teacheck. Este servicio trata de convertir todos los datos proporcionados por el sistema de la universidad cliente al formato adecuado que necesita nuestro sistema.
- **Servicio Mensajería:** El servicio de mensajería trata transmitir mensajes desde un dispositivo que controla asistencia de alumnos a nuestro sistema. La implementación hace el equipo de Teacheck. Este servicio es adicional y no es obligatorio para el funcionamiento del sistema.
- **Servicio Base de Datos:** Este es el servicio central que se comunica con la base de datos de Teacheck. Se encarga de todas la transacciones y consultas del sistema.

2.3 Diseño centrado en el usuario

Una buena aplicación necesita un buen diseño, pero para que sea bueno no es suficiente hacerlo bonito, también es necesario hacerlo útil. Para asegurarnos de esto, hemos seguido lo que se llama Diseño centrado en el usuario. DCU es una metodología enfocada a las necesidades reales del usuario.

2.3.1 Contexto de uso

Primero debemos analizar los diferentes tipos de usuario que la aplicación tiene y como interactúan ellos con la aplicación.

Algunas partes de la aplicación pueden, y probablemente serán complejas, pero eso no debería ser un problema. La complejidad no siempre significa dificultad, a veces las tareas complejas se pueden realizar fácilmente. Tiene que ser una herramienta poderosa, que nos aporte mucho, pero a la vez, que sea fácil de utilizar.

Entorno Esta aplicación está diseñada para Universidades y son los los integrantes de esa universidad los que la utilizarán. Debe ser una herramienta útil que nos proporcione la información necesaria para conseguir los objetivos. Por lo tanto, la eficiencia es mucho más importante que la estética.

2.3.2 Usuarios

- **Alumnos:** Es una aplicación en la que los alumnos no pasarán mucho tiempo ya que no tendrá muchas funcionalidades. Por una lado, los alumnos podrán ver su perfil (foto, datos personales, estadísticas generales,...). Por otro lado, podrán ver el estado en el que están, es decir, estadísticas e información más ecisa tales como estado del curso, sus notas, asistencia, entregables, etc. Y por último, podrán realizar las encuestas semanales. Estas encuestas son rápidas y sencillas pero que serán muy útiles a la hora de analizar a los alumnos.

  Cada año, nuevos alumnos comenzarán a utilizar la aplicación y otros que ya la utilizaban dejarán de hacerlo al terminar sus estudios. Por lo tanto, tiene que ser una aplicación fácil de aprender y de utilizar.

- **Profesores:** Probablemente, son los usuarios que más utilizarán la aplicación. Estos, tendrán disponibles tres apartados. En el primero, podrán ver el estado de todos los alumnos de cada curso donde da clase. En el segundo, tendrá el apartado de alarmas donde verá que alumnos suyos están en peligro o necesitan un toque de atención. Y por último, tendrá el apartado de las estadísticas generales, es decir, se le mostraran los resultados generales de las encuestas a nivel de clase.

- **Coordinadores del curso:** Los coordinadores del curso, al igual que los profesores, también pasarán tiempo con esta aplicación. Y al igual que los profesores, podrán ver el estado de los alumnos de el curso o los cursos que coordina, el apartado de las alarmas y los resultados generales de las encuestas semanales. Pero a diferencia de los profesores, las alarmas que se le notifiquen no serán solo a nivel de alumno. También se le avisarán de los alarmas a nivel de clase.

2.3.3 Requisitos

Requisitos de negocio

GENERAL

- La aplicación permitirá hacer diferentes cosas al usuario según el rol asignado.
- La aplicación debe estar protegida por un login.

- La aplicación guardará los datos de manera protegida ya que maneja datos delicados.

ALUMNOS

- La aplicación tiene que ser capaz de visualizar estadísticas de los alumnos tales como el estado, notas, asistencia y entregables.
- La aplicación debe ser capaz de mostrar el perfil al usuario con su foto y datos personales.
- La aplicación permitirá rellenar a los alumnos encuestas semanales.
- La aplicación hará análisis periódicos de los alumnos con las fechas definidas anteriormente.

PROFESORES

- La aplicación permitirá a los profesores ver el el estado de los alumnos de las clases donde imparte clases.
- La aplicación permitirá a los profesores ver el apartado de alarmas donde verá todos los alumnos a los que le ha saltado la alarma.
- La aplicación mostrará a los profesores los resultados generales de las encuestas por cada asignatura que enseña.
- La aplicación deberá enviar dichas alarmas a los profesores vía email.

COORDINADORES DEL CURSO

- La aplicación permitirá a los coordinadores ver el el estado de los alumnos del curso que coordina.
- La aplicación permitirá a los coordinadores ver el apartado de alarmas donde verá todos los alumnos a los que le ha saltado la alarma y todas las alarmas a nivel de clase.

- La aplicación mostrará a los coordinadores los resultados generales de las encuestas por cada curso que coordina.
- La aplicación deberá enviar dichas alarmas a los coordinadores vía email.

Requisitos de usuario

ALUMNOS

- Los alumnos tienen que ser capaces de ver su estado del curso mediante tablas, listas o dasboards.
- Los alumnos tienen que ser capaces de acceder a su perfil mediante un simple botón situado a la izquierda.
- Los alumnos tienen que ser capaces de rellenar encuestas semanales mediante formularios.

PROFESORES

- Los profesores tienen que ser capaces de ver el estado de sus alumnos a través de listas, tablas de datos o dashboards.
- Los profesores tienen que ser capaces de ver las alarmas de dichos alumnos mediante listas.
- Los profesores tienen que ser capaces de ver los resultados de las encuestas mediante tablas o gráficos.
- Los profesores tienen que ser capaces de acceder a su perfil mediante un simple botón situado a la izquierda.

COORDINADORES DEL CURSO

- Los coordinadores tienen que ser capaces de ver el estado de sus alumnos a través de listas, tablas de datos o dashboards.
- Los coordinadores tienen que ser capaces de ver las alarmas de dichos alumnos mediante listas.

- Los coordinadores tienen que ser capaces de ver los resultados de las encuestas mediante tablas o gráficos.
- Los coordinadores tienen que ser capaces de acceder a su perfil mediante un simple botón situado a la izquierda.

Requisitos funcionales

- API de Highcharts. Esto se necesitará para crear gráficos con altos detalles de visualización de datos.
- API's de Vertx. Se utilizarán para la construcción del sistema.
- La aplicación debe desarrollarse en el lenguaje de programación Java.
- Se va a utilizar Docker para crear una imagen de la aplicación para implementarla en un servidor remoto.
- La aplicación tiene que estar desarrollada en una plataforma multilenguaje.

Requisitos de calidad de servicio

- La aplicación estará disponible 24/7.
- La aplicación debe ser escalable, es decir, al introducir nuevas asignaturas no debe influir en el modelo de la arquitectura.
- El sistema debe ser tolerante a fallos, incluso si ocurre un problema de hardware, el sistema debe seguir funcionando como se espera.
- La seguridad de la aplicación debe incluir autorización, autenticación, seguridad de acceso a datos y acceso seguro para el sistema de implementación, así como su administración.
- El sistema debe ser mantenible, monitoreado y actualizado cuando sea necesario.

2.4 Casos de uso

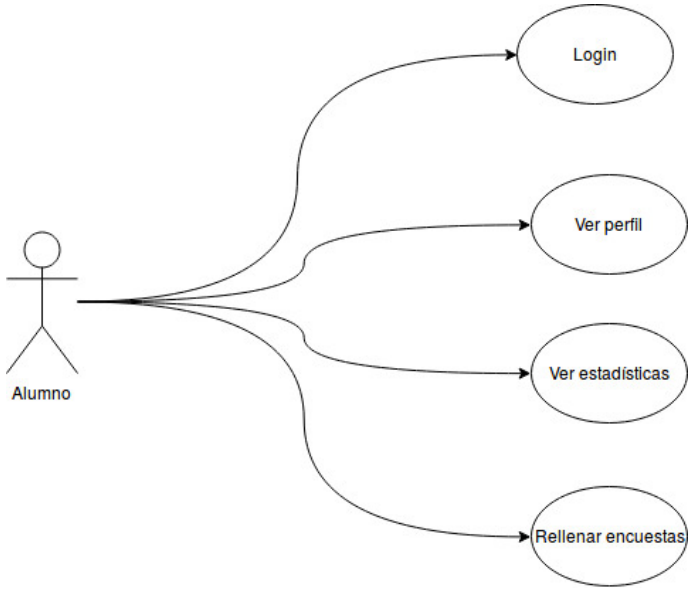


Figure 2.4: Casos de uso - Alumno

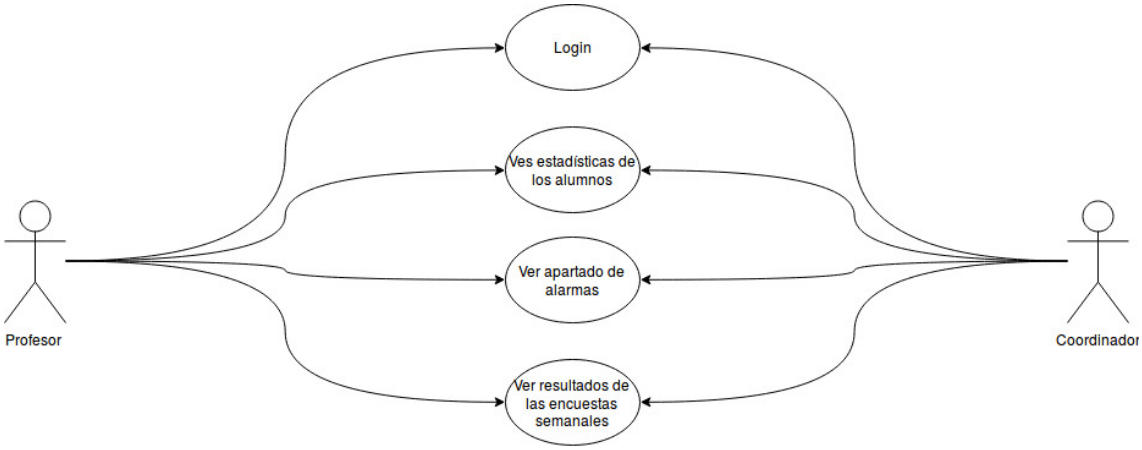


Figure 2.5: Casos de uso - Profesor y coordinador

3. CHAPTER

Organización del Proyecto

3.1 Planning Inicial

Antes de empezar a programar hemos decidido tanto pensar cómo planificar cómo vamos a trabajar estas semanas.

Para comenzar hemos decidido utilizar Github. Primero para poder codificar y desarrollar distintas partes del proyecto de manera ordenada. No crearemos diferentes ramas sino que haremos uso de las "Issues" que nos facilitan saber que se ha cambiado o creado en cada commit que vayamos a ejecutar. También usaremos Github para organizar las diferentes tareas que tenemos que llevar a cabo, haciendo uso de su 'Github project board'. Aquí se podrán visualizar las distintas tareas que se tienen que hacer, están en desarrollo y se encuentran ya acabadas.

Luego haremos uso de docker que permite que una aplicación y todas sus dependencias se empaqueten en un contenedor y puedan ejecutarse sin tener en cuenta el sistema operativo de la máquina en la que se encuentra el docker. De esta manera evitaremos el tan conocido problema de que alguien tenga problemas al ejecutar algo desarrollado en otra máquina.

También usaremos maven y vertx, un framework que nos permite desplegar microservicios apoyándose en programación reactiva. Lo hemos elegido tras hacer varias pruebas y ver lo sencillo que era su uso.

Por último haremos una reunión semanal con nuestro tutor para asegurar que seguimos un rumbo correcto.

3.2 Entorno de Desarrollo

3.2.1 Entorno Virtual de Desarrollo Docker

El entorno de desarrollo de la aplicación debe ser uniforme para todos los desarrolladores. Es una tarea difícil ya que siempre hay diferencias de sistemas operativos y cada máquina es unica. Pero hay una solución que Teacheck utilizará tanto para el entorno de desarrollo como para el de producción y servidor de testeo, Docker.

Docker es una herramienta que permite la "contenerización" de sistemas completos (una especie de virtual machine aunque no lo es). Los containers (así llamados dichos sistemas) creados por Docker se comunican directamente con el kernel del sistema operativo. Eso permite que un entorno sea el mismo en cualquier otra máquina independiente de su sistema.

Dicho esto el valor que nos aporta Docker es el entorno uniforme entre todos los desarrolladores así como el entorno de producción y testeo. Además de facilitar las dependencias de desarrollo como base de datos para testeo por ejemplo. Docker permite desplegar entornos completos sin tener que instalar software adicional.

El entorno funciona con una serie de containers desplegados en una red virtual de Docker con toda la configuración necesaria para que funcionen correctamente. Para ese múltiple despliegue y comunicación entre esos containers en una red virtual se utiliza una extensión de Docker, docker-compose. La idea por detrás de esta herramienta es simplemente automatizar ciertos aspectos de Docker para que sea más productivo. La herramienta docker-compose tiene la capacidad de desplegar un entorno compuesto de varios containers dentro de una red virtual donde pueden comunicarse entre sí. Se podría hacer lo mismo utilizando solamente Docker pero sería algo muy manual y con altas probabilidades de error.

Por lo tanto el desarrollo de todo los servicios y componentes que componen el sistema de Teacheck utiliza esta herramienta. Cada componente tiene su desarrollo separado de los demás ya que cada uno es un subsistema independiente. A seguir se detalla el entorno de desarrollo de la aplicación web, un ejemplo de cómo se lleva a cabo la configuración y despliegue de un entorno. Los entornos de los demás subsistemas sólo se diferencian en la configuración pero la metodología es la misma.

El contenedor de la aplicación web. Este contenedor es un sistema operativo basado en Alpine Linux y tendrá Maven con la versión más reciente y además con la versión 8 del openjdk de Java. En este sistema se ubicará todo el código en una carpeta "/app". Esta carpeta estará mapeada al sistema del host, es decir, el sistema operativo que hospeda el contenedor. Dicha carpeta recibirá los cambios hechos en el host y así el contenedor se actualiza con cada nueva iteración en el código. Esto nos permite que todo tipo comando de Maven, por ejemplo los builds y testeos, se ejecuten dentro de ese sistema, totalmente aislado del sistema host. Para llevar a cabo la productividad aun más todavía el contenedor ejecutara un bash script para mantener la aplicación en marcha "escuchando" cualquier cambio en el código. Si algún cambio ocurre el script desplegará la aplicación otra vez con el código fuente actualizado. En realidad el contenedor de la aplicación moriría si no fuera por el script ya que los contenedores de Docker solo viven hasta que el comando que lo inicializo termina su ejecución.Por eso el sistema del contenedor al inicializarse ejecutar dicho script para arrancar la aplicación, la cual a su vez pone en marcha un servidor HTTP escuchando en el puerto 8080 del contenedor.

El siguiente contenedor es la base de datos. La base de datos, en este caso PostgreSQL, también correrá en un sistema operativo Linux. Simplemente para testeo durante el desarrollo.Se utilizará la configuración por defecto del contenedor y algunas variables de entorno adicionales para declarar el nombre de la base datos y la contraseña. Esto es por un concepto muy importante en respecto a los contenedores de Docker. Todos los contenedores son efímeros, una vez que el entorno es terminado todos los cambios hechos al contenedor son borrados y si se intenta desplegar el entorno una vez más pero este último tendría la configuración por defecto. Por lo que permite que se cometan errores ya que si algo crítico ocurre con la base de datos por ejemplo, con simplemente terminar y desplegar otra vez el contenedor se solucionaría el problema.

Existen otros dos contenedores, Nginx y Keycloak, el primero es un reverse proxy para la aplicación web y el segundo es un Identity Provider (IDP) para gestionar el control de acceso y entidades para la aplicación. Utilizan configuraciones por defecto recomendadas para el desarrollo.

Todo esto es transparente para el desarrollo una vez el proceso de configuración del entorno esté completa, lo que se hace una vez al principio de un nuevo proyecto. En el decorrer del desarrollo de la aplicación el desarrollador solo utilizará dos comandos básicos para el despliegue y visualización de los logs de cada contenedor si así se desea.

Esta sería la definición del entorno de desarrollo para ejecución con docker-compose:

```
services:
  app:
    image: maven:3-jdk-8-alpine
    ports:
      - 3000:8080
    volumes:
      - ./app:/app:delegated
      - ~/.m2:/root/.m2:delegated
    working_dir: "/app"
    command: ["sh", "redeploy.sh"]
  postgres:
    image: postgres:9.6
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: keycloak
      POSTGRES_USER: keycloak
      POSTGRES_PASSWORD: password
  keycloak:
    image: jboss/keycloak:6.0.1
    environment:
      DB_VENDOR: POSTGRES
      DB_ADDR: postgres
      DB_DATABASE: keycloak
      DB_USER: keycloak
      DB_SCHEMA: public
      DB_PASSWORD: password
      KEYCLOAK_USER: admin
      KEYCLOAK_PASSWORD: admin
    ports:
      - 8080:8080
      - 8443:8443
    depends_on:
      - postgres
  nginx:
    image: nginx:1.12.2
    ports:
      - 80:80
    depends_on:
      - app
    volumes:
      - ./proxy/conf.d:/etc/nginx/conf.d:ro
```

Figure 3.1: Esquema del entorno con los diferentes contenedores que son necesarios para llevar a cabo el desarrollo de la aplicación

Aquí un pequeño esquema del entorno con los diferentes contenedores que son necesarios para llevar a cabo el desarrollo de la aplicación.

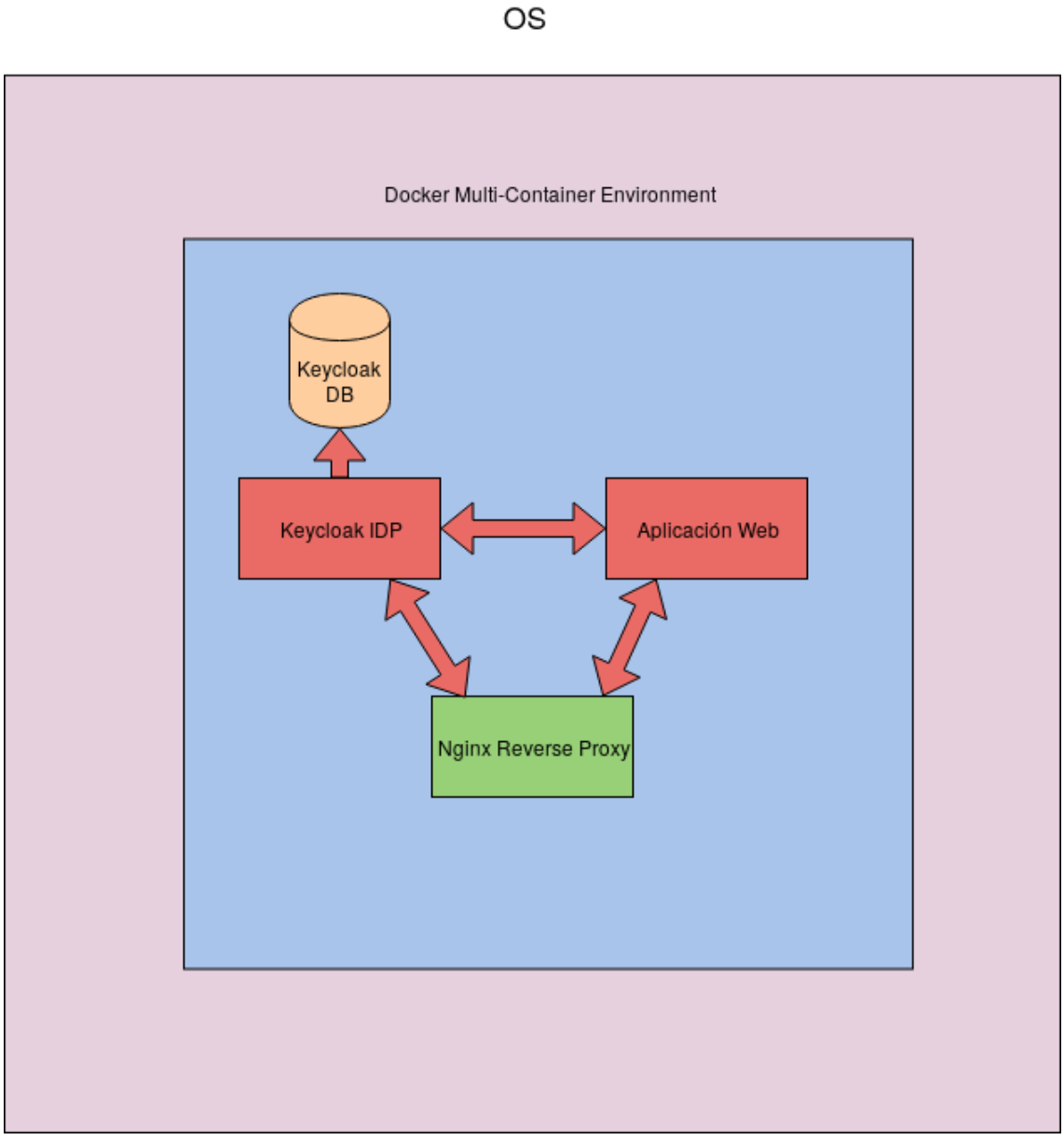


Figure 3.2: Entorno de Desarrollo en contenedores

3.2.2 Control de Versiones

Cada proyecto de desarrollo de software tiene que mantener un control de las versiones del producto. El control de versiones es muy importante y conveniente para los desarrol-

ladores. Teacheck utilizará Git.

Git es uno de los sistemas más populares en el mundo para el control de versiones. Es distribuido por lo que nos da la capacidad de que cada desarrollador tenga un clon del repositorio completo en su propia máquina. También se utilizará la plataforma en la nube central de Git, GitHub. Allí se ubicará el repositorio remoto de la aplicación donde se subirán los cambios periódicamente al código.

La organización entre los desarrolladores es muy importante a la hora de llevar a cabo la construcción de un sistema. GitHub proporciona un sistema de issues donde permite crear diferentes tareas relacionadas al repositorio. Cada issue tiene un identificador y se le puede añadir documentación, comentarios y asignarlas a colaboradores del proyecto. Teniendo en cuenta este sistema se plantea el siguiente ciclo de desarrollo para nuevas iteraciones:

- Para cada nueva iteración a desarrollar se debe crear un nuevo issue y escribir una dedicada documentación sobre lo que se quiere hacer.
- Asignarlas a los responsables de la tarea
- Si es necesario determinar qué tipo de issue es:
  - Enhancement
  - Bug
  - Task

Esto trae beneficio a todos ya que se mantiene claro y organizado las tareas que se deben llevar a cabo.

3.2.3 Trunk Based Development

Se aplicará en este proyecto la metodología Trunk Based Development. No se utilizarán ramas adicionales para el desarrollo si no la rama principal master. Esto nos permitirá tener más responsabilidad a la hora de subir actualizaciones a la rama principal y siempre intentando mejora la calidad del código. Además proporciona más productividad y es perfecto para proyectos pequeños.

Permite una mejor visibilidad y historico de actualizaciones al código. Mantiene el foco siempre claro en lo que se está desarrollando ya que también cada commit tendrá consigo la referencia a que issue pertenece. Así todos los desarrolladores pueden identificar fácilmente cada commit y saber que tipo de funcionalidad se está desarrollando.

3.2.4 Organización de tareas GitHub

La plataforma GitHub tiene un apartado en cada repositorio llamado Projects. En este apartado el sistema de GitHub permite crear Kanban Boards para coordinar tareas relacionadas al repositorio en este caso al software que está en desarrollo. Se aprovechará esta funcionalidad para coordinar todos los issues creados en el repositorio. Eso es, GitHub automáticamente detecta los issues creados en el repositorio y permite utilizarlas como cards para el panel de tareas.Otro punto positivo es la disposición de toda la documentación del issue en el panel, es decir, con apenas un click se puede obtener todos los detalles de la tarea: a quien está asignada, documentación, que tipo de issue es y etc.Para clasificar en qué estado se encuentra cada issue existen los siguientes apartados:

- To do
- In progress
- Done!

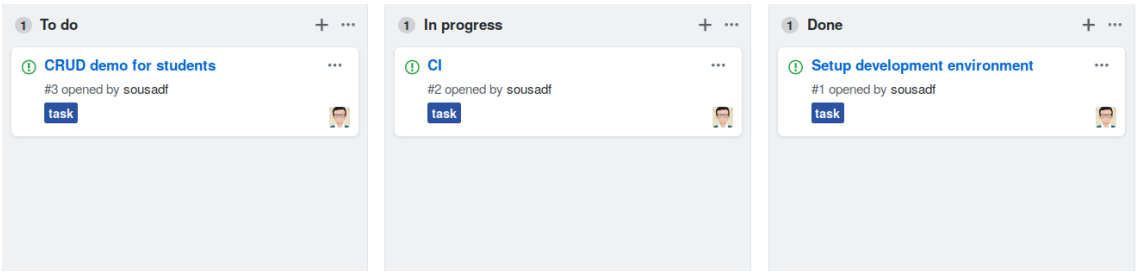


Figure 3.3: Kanban Board

Esto servirá para mantener las tareas organizadas y además de ofrecer un historial de todo lo que se ha hecho en el decorrer del proyecto.

3.2.5 Documentación del código

Un buen código es aquel que habla por sí solo y no necesita explicaciones adicionales. Pero mantener el código claro y conciso es una tarea difícil y en muchos casos imposible. Cada sistema tiende a profundizar en complejidad según más funcionalidades se añadan. Cuanto mayor el sistema más complejo será. Por lo tanto la buena estructuración y documentación de partes complejas del sistema es importante para el entendimiento del que intenta moverse por el mismo.

Teacheck utilizará Doxygen, que es la de facto herramienta estándar para generar documentación desde un código fuente de Java. La documentación se realizará apenas sobre funciones y partes complejas del sistema. Debe ser clara y enfocada en la función en cuestión siguiendo los estándares de Doxygen para escribir documentación en Java.

3.2.6 Integración Continua

Uno de los puntos importantes cuando se trata del desarrollo de software es asegurar que todo el análisis en respecto al código sea realizado. Ese análisis está compuesto de diversas operaciones que verifican calidad del código producido. Esas operaciones son los tests unitarios, el build, formatos, distribución del código y etc. Con integración continua se puede asegurar que a cada push de cambios al repositorio remoto pase por el sistema de control de calidad del código. El valor que aporta este proceso es evidente y además de ser un sistema automatizado. Mantener el flujo de desarrollo eficiente, productivo y al mismo tiempo asegurando la calidad del producto en desarrollo.

Para ello se decidió utilizar un sistema de integración continua. Existen muchos sistemas, pero el elegido es Travis CI por su facilidad de integración con GitHub y empleo.

El proceso inicial de implementación del sistema de Integración Continua es muy simple. Travis permite mantener un script con la descripción de cómo se deberán realizar los controles de calidad. En el caso de Teacheck, siguiendo la filosofía de mantener el entorno uniforme, todo el pipeline por donde pasará el código se ejecutará en un entorno de Docker. Ese entorno será el mismo que el de desarrollo. En dicho entorno se realizarán los testeos unitarios y de integración, el coverage test y el packaging de la aplicación en

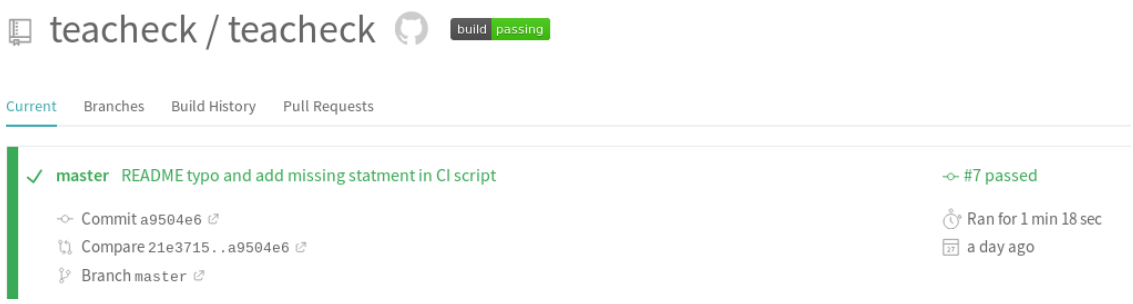


Figure 3.4: Pipeline status example

un fat jar (Un archivo JAR con todas la dependencias) con la nueva versión. Por último se creará una imagen a partir del JAR creado y se publicará en la plataforma de Docker Hub (registro de imágenes Docker).

3.2.7 Estándar de Codificación

Nomenclatura Java por defecto a la hora de nombrar clases, variables, constantes, etc. es una mezcla entre la nomenclatura tradicional en Inglés y la nomenclatura funcional adoptada. Es decir que por estandarización se puede escribir en Inglés y se mantendrá así por convenio, casos como insert, update, delete, create, retrieve, list, set, get, newInstance, etc.

Para la parte funcional se utilizará castellano. Así pues los métodos podrán tener la mezcla dicha antes del Inglés. Ejemplos como getListAlumnos o updateAlumno.

Paquetes Así como en la mayoría de los estándares de Java los nombres de los paquetes deberán ser escritos en minúsculas y libre de caracteres especiales. Como este proyecto utiliza Maven como el build tool la estructuración inicial de los paquetes es la siguiente:

- java
  - main
    - com.teacheck (paquete base)
  - test
    - com.teacheck (paquete base)

Como se puede ver en la estructura el paquete base tiene como nombre com.teacheck. Este paquete no definirá ninguna clase.

A partir de este paquete base se definirá la estructura en árbol de los paquetes.

- business
  - dao
  - domain
  - service
  - helper
  - exception
- util
- web
  - controller
  - model
  - view
- common

Todo que sea común en muchas partes del sistema se deberán ubicar en la carpeta common.

Nombre de Interfaces

Los nombres de las interfaces tendrán como sufijo una I. El nombre puede esta compuesto de multiples palabras terminando en el sufijo concretado anteriormente. La primera letra de cada palabra debe estar escrita en mayúscula(CamelCase). Se debe evitar nombres abreviados lo que por su vez dificulta el entendimiento de la interfaz.

Ejemplo: UserManagerI

Nombre de Clases

Asi como las interfaces las clases deben seguir el convenio de nombres CamelCase. Es decir la primera letra de una palabra siempre en mayúsculas. Los nombre tienen que ser descriptivos y entendibles sin abreviaciones no comunes. Algunas de las abreviaciones estándares como DAO, DTO,URL, JDBC, etc. son permitidas.

Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase). No se permiten caracteres especiales.

El nombre debe de ser descriptivo y en principio la longitud del mismo no es importante.

Variables

Los nombres de las variables utilizan las mismas reglas que los métodos con excepción de la longitud. Intentar no utilizar abreviaciones. Con excepciones en casos en el cual la abreviación es común y evidente.

Evitar caracteres especiales. Nombres sin ningún tipo de significado como i o j. Pero por ejemplo se pueden utilizar en bucles como el for.

El primer bucle siempre tendrá como variable i como iterador.

Constantes

Las constantes de clase deberán escribirse todo en mayúsculas con las palabras separadas por subrayados *underscore*. Todas será declaradas como public static final. Con excepción de constantes privadas a la clase.

Estilo de codificación - Comentarios

Los comentarios se pueden utilizar para añadir información a alguna clase o método. Dichos comentarios son para el mejor entendimiento del que lee sobre el diseño de la implementación. Este tipo de documentación solo es aconsejable en partes donde no está claro para qué sirve o lo que hace.

Estilo de codificación - Documentación

La documentación de cada clase es obligatorio. Métodos y constantes a su vez deberán disponer de comentarios de documentación cuando sea necesario. Es decir en partes complejas donde es difícil comprender lo que hacer.

Para cada clase y interfaz se debe haber una descripción genérica sobre su propósito y responsabilidad. Además también llevar el nombre del autor o autores de clase / interfaz.La documentación deberá ser escrita en tercera persona

Se deberán utilizar tags para documentar parámetros, lo que devuelve cada método, excepciones que pueden ocurrir, etc. Aquí el orden en el cual se deben documentar dichos tags:

- **@param** La descripción de cada parámetro debe definir el tipo del mismo seguido que define el parámetro
- **@return** este tag no es necesario para métodos que tienen como tipo de retorno el void.
- **@throws** Descripción breve de la excepción. Lo que pueda causar la excepción.
- **@link** Link para complementar la documentación con alguna otra explicación o relación al documentado.

Se debe evitar el uso excesivo de link para no llenar la documentación de enlaces.

Se puede utilizar el atributo <code> para palabras reservadas de Java como nombres de clases, interfaces, propriedades, etc.

Se debe evitar el uso excesivo de link para no llenar la documentación de enlaces.

Se puede utilizar el atributo <code> para palabras reservadas de Java como nombres de clases, interfaces, propriedades, etc.

Estilo de codificación - Documentación

Para La declaración de variables se utilizará una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propriedades de un objeto.

la codificación correcta sería:

public static Integer entero = new Integer(0);

4. CHAPTER

Lo de IA

4.1 Introducción

El núcleo de la aplicación reside en una máquina la cual tiene tres diferentes objetivos. El primero, predecir el rendimiento de un alumno a lo largo del curso, creando alarmas cuando sean necesarias. El segundo, poder analizar y descubrir los detonantes de las posibles alarmas, ya que sirve de poco saber de la existencia de un problema sin conocer las razones. Y por último, poder a su vez predecir cuando el rendimiento de una clase a decaído, y al igual que con un alumno, poder crear una alarma de dicho problema.

Dichos objetivos se han logrado haciendo uso del aprendizaje automático, una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. De esta manera, hemos generado modelos, extraído correlaciones y desentrañado clasificadores de datos como el bayesiano ingenuo(1). Pero antes de entrar en detalle y ver cómo se ha analizado, diseñado e implementado todo lo mencionado, vamos a dar un paso atrás, ya que para poder desarrollar todo lo mencionado, se necesitan datos.

Dichos datos son necesarios tanto para crear los modelos que se usarán como predictores, como a su vez para luego testearlos y hacer funcionar nuestra aplicación. Además, no son datos generados aleatoriamente, si no que tienen que seguir una lógica, una lógica que heredará el predictor inteligente, y a su vez toda la aplicación. Por lo tanto, se ha

realizado un análisis de los datos que se han querido generar, y del método utilizado para su generación.

4.2 Datos

4.2.1 Variables

Tal y como se acaba de mencionar, los datos cuentan con una gran relevancia, y debido a eso, cada dato o atributo se analiza, define y genera de forma individual. Dichos datos o atributos pertenecen a cada asignatura de un alumno, esto es, contamos con nueve atributos por asignatura, siendo cinco las asignaturas por semana y diez las semanas por curso. Los atributos son los siguientes:

- **Notas de competencias:** El primer atributo para analizar serán las notas conseguidas por los alumnos en los diferentes puntos de control de cada competencia. Este atributo será un decimal que nos ayudará a saber el desempeño del alumno en cuanto a un temario y tendrá un rango del 0 al 10.
- **Asistencia al curso:**

  El segundo atributo que se tendrá en cuenta a la hora de realizar el análisis será la asistencia de un alumno a una asignatura a lo largo de la semana. Esta característica creemos que es importante ya que, aunque para cada universidad influya de una forma diferente en cuanto al desempeño del alumnado, es un atributo directamente relacionado con el entendimiento de las clases impartidas. Este dato será un porcentaje en relación con las horas de clase impartidas en un semana de una asignatura y la asistencia de un alumno a las mismas.
- **Motivación del alumno:**

  Este tercer atributo se ha decidido que sea opcional, esto es, en caso de que el profesor no vea nada excepcional en un alumno, no tendrá la necesidad de aportar ninguna información sobre el o ella. Está pensado para aquellas situaciones en las que el personal docente observe algún tipo de variación en la motivación del alumno. Por ello, será un número con un rango del 1 al 5 en el que un 1 sería una motivación nula en clase y 5 un comportamiento ejemplar del alumno.

- **Atención del alumno:**

  Además, en cuanto a la información recogida por el profesor, el cuarto atributo que tendremos en cuenta será la atención de un alumno en una asignatura a lo largo de la semana. Se ha comprobado que, aunque el atributo de asistencia esté cerca del 100% si un alumno no presta atención en clase, las probabilidades de que ese sujeto no esté entendiendo el contenido de la asignatura son peligrosamente altas. El formato del dato será idéntico a la motivación.

- **Entregables o trabajos de la asignatura:**

  Este atributo estará relacionado con los diferentes trabajos a entregar en una asignatura en concreto. Se decidió tener en cuenta esta característica ya que los entregables están relacionados con el contenido práctico de una asignatura, y, por tanto, en caso de haber realizado el trabajo, las probabilidades de que un alumno se desempeñe correctamente en esa asignatura son muy altas. Esta característica solo nos dirá si se ha realizado dicha práctica, por tanto, será un booleano.

  - Nota de los entregables o trabajos de la asignatura: Este atributo únicamente será rellenado en caso de que el atributo anterior sea verdadero y será un número con un rango del 0 al 10.

- **Motivación:**

  Al igual que el personal docente puede saber la motivación de un alumno en clase, el propio alumno también será capaz de realizar una encuesta y decirnos si realmente está motivado en una asignatura o no. Este dato será ingresado desde la encuesta semanal que el alumnado tendrá la oportunidad de realizar al finalizar la semana. Los datos ingresados serán idénticos a la motivación ingresada por el personal docente. Está comprobado que dado el anonimato desde el que se ingresará esta información, el alumnado podrá sincerarse y hacer una valiosa aportación al sistema.

- **Atención:**

Como el atributo anterior, en este caso, el alumnado será capaz de ingresar este tipo de información desde la encuesta al finalizar la semana. Nos interesa tanto la atención que un alumno pueda tener en clase desde el punto de vista del profesor y del alumno, ya que, en algunos casos, pueda parecer que un alumno no esté prestando atención y realmente si lo esté haciendo y viceversa.

- **Horas aplicadas a la asignatura fuera de clase:**

Por último, sabemos que la universidad requiere de una cantidad de horas de estudio fuera de la misma. Es un número subjetivo, ya que cada alumno necesitará de diferente tiempo para entender una asignatura. Pero gracias a las medias que podamos conseguir de una clase entera, sabremos si un alumno holgazanea al llegar a casa o realmente repasa el contenido dado en clase.

4.2.2 Horizonte de análisis

Haciendo uso de estos datos, tal y como se ha mencionado anteriormente, se predice el rendimiento de cada alumno en cada una de las asignaturas en los que se encuentra matriculado durante el curso. Dicho curso cuenta con diez semanas, y cada semana con un modelo diferente. Esta división se debe a que el número de datos en las primeras semanas y en las últimas varía considerablemente. Por lo tanto, no todos los modelos difieren entre sí, pero sí que hay una evolución en los modelos a medida que avanza el curso.

Los datos o conclusiones que sacaremos serán los siguientes. Primero, en cuanto al rendimiento del alumno, se determinará por asignatura si el alumno requiere de una alarma o no. Luego, se detalla el impacto de cada atributo a la hora de decidir la alarma, lo cual se utiliza para sacar el detonante de la misma. Por último, usamos las correlaciones y el número de alarmas para determinar y generar las alarmas de clase. Más adelante se detalla el procedimiento de cada una de ellas.

4.2.3 Creación datos

Una vez definidas las variables hemos creado los datos que se usarán para diseñar y crear los modelos. La creación de dichos datos no ha sido aleatoria, ya que aun siendo

una simulación se requerían datos que cumpliesen una lógica para de esta manera poder crear modelos que cumpliesen con su función.

Dicho esto, los datos se han generado utilizando Excel(2), básicamente debido a su facilidad de generar muchos datos haciendo uso de sus macros. Por ello, se ha utilizado una macro para cada variable, y estas se han generado siguiendo una distribución normal que contaba con una desviación típica. De esta manera, generamos cientos de datos aleatorios que seguían una lógica definida por nosotros, la cual definimos tras estudiar y comparar datos reales.

Hablando de números, se han creado datos de 300 alumnos para diez semanas de lo que podría ser un curso, contando cada semana con cinco asignaturas. En las dos primeras semanas tanto la nota del examen como el entregable están vacías, y de la tercera a la quinta se añaden datos tanto al entregable como a su nota. Esto se hizo con el objetivo de tener datos más reales, ya que en las primeras semanas de un curso real no se suelen contar con estos datos.

Una vez generados todos los datos para las diez semanas, comenzamos a diseñar lo que sería la función que más tarde generaría el target, esto es, lo que definiría si el alumno está necesitado de una alarma o no. Para ello, dimos un peso a cada variable, creamos la función, y después de insertar la misma en una macro de Excel, generamos el target.

Y de esta manera, haciendo uso de una distribución normal seguida de una desviación típica y añadiendo la función diseñada por nosotros, generamos los datos de forma lógica dándoles el sentido que buscábamos y necesitábamos para poder crear nuestros modelos.

(aquí se puede insertar foto o tabla o lo que sea con las distribuciones que hemos seguido. Tmb para la función.)

4.2.4 Creación de modelos

Tras generar los datos, continuamos con la creación de los modelos anteriormente mencionados, esto es, uno por semana.

Los dos primeros modelos no cuentan ni con la nota del examen ni con el entregable, y los siguientes tres siguen sin contar con la nota del examen. Una vez llegados a la quinta semana, se cuenta con todos los datos. Es cierto que no son diez modelos únicos, pero la decisión de crear todos ellos se basa en el hecho de que en un caso real se requeriría un modelo personalizado e individual por semana.

Para generar los modelos se utilizó el clasificador bayesiano ingenuo, debido a que de las pocas opciones disponibles, siendo la diferencia a la hora de predecir tan insignificante, era con la que más familiarizada estábamos. Para la creación de dichos modelos, utilizamos el 80% del dataset antes mencionado, esto es, 240 de los 300 alumnos. El 20% restante se usó para el testeo de los modelos, y más tarde para insertarlos en la base de datos a modo de nuevos alumnos. Los resultados fueron positivos, el porcentaje de acierto rondaba el 85-90%, lo cual cumplía con las expectativas. Además de ello, se consiguen las correlaciones para después poder realizar un análisis y determinar si se requiere de una alarma de clase o no. Si el número de alarmas en una clase supera el 50%, saltará una alarma a nivel de clase y los detonantes serán las correlaciones antes mencionadas.

4.3 Enseñanza automática

4.3.1 Predicciones y correlaciones

Con el objetivo de lograr resultados haciendo uso de los modelos previamente mencionados, se decidió crear un servicio web en R, el cual una vez está activo y escuchando, recibe información en formato JSON relativa a 60 estudiantes por cada una de las 5 asignaturas en una semana.

Se requiere que unos de los datos que va a ser enviado a este servicio junto con todos los atributos del alumno, sea el número de la semana, para así poder saber cuál es el modelo a aplicar en cada semana.

Por lo tanto, el procedimiento una semana cualquiera, será el siguiente: Para empezar, la información previamente definida del alumnado debe ser enviada al servicio web en marcha. Luego, esta información es formateada dependiendo del tipo de dato que tenga cada atributo, ya que para poder realizar predicciones y correlaciones todos los atributos deben ser numéricos y los valores nulos necesitan ser traducidos a un formato para que, a la hora de predecir, se tengan en cuenta como el valor que son, esto es, un 0. Una vez la información pueda ser interpretada por el predictor, se utiliza el modelo generado para la semana de la que provenga la información, y se predice la clase, nuestro caso, es un atributo con dos posibles valores “Alarma” y “No Alarma”.

Una vez tenemos una columna con esta información, es el momento de sacar, a nivel general, esto es de los 60 alumnos, el peso que ha tenido cada atributo a la hora de tomar la decisión. Para ello simplemente son comparados cada atributo de todos los alumnos con

la nueva columna de predicciones que acabamos de generar. Estos llamados coeficientes de correlaciones, nos indicarán globalmente qué atributo ha sido decisivo a la hora de indicar la alarma y, por tanto, todos y cada uno de los atributos del alumnado tendrán un porcentaje con su nivel de influencia. Estos coeficientes de correlaciones pueden ser calculados con diferentes métodos: el coeficiente de correlación de concordancia Kendall, el coeficiente de correlación de Pearson y el coeficiente de correlación de Spearman. En nuestro caso, calcularemos el coeficiente de correlación de Kendall (8) ya que contamos con ambos tipos de valores, ordinales e intervalos. Cada semana, tanto las predicciones como las correlaciones variarán, ya que teniendo en cuenta el modelo se generarán X alarmas, y en función de estas alarmas, las relativas correlaciones.

Desde una perspectiva más general, por tanto, a este script se le envía un mensaje en formato JSON relativo a 60 alumnos con toda la información recopilada en una semana por cada asignatura, exceptuando la clase, y nos devolverá un resultado con la clase y el id de cada alumno. Además, en otra lista, los 9 atributos del alumnado cada uno con el valor de su correlación respecto a la clase.

4.3.2 Clasificador bayesiano ingenuo

Una vez conseguidos los resultados anteriores, se estudió la posibilidad de encontrar el motivo por el cual a cada alumno se le había sido asignado un valor de la clase u otro, esto es, el motivo por el cual una alarma había saltado o no. Gracias a las correlaciones pudimos saber qué tendencia seguía una clase por asignatura, pero no cómo el modelo había llegado a asignar cada valor a la clase.

Por lo tanto, después de analizar y definir cómo funcionaba el método del bayesiano ingenuo que usamos para generar los modelos y más tarde realizar las predicciones, se empezó a aplicarlo al dataset creado para la generación de los modelos y así poder conocer el impacto individual de cada atributo a la hora de decidir la clase.

Antes de explicar cómo se aplicó dicho método bayesiano al dataset utilizado en este proyecto, se realizará un pequeño resumen del funcionamiento del clasificador bayesiano ingenuo. Este clasificador, tiene en cuenta cada posible valor que puede tener un atributo, así pues, en nuestro caso, por ejemplo, la atención es un atributo cuyos valores posibles son un numérico del 1 al 5. Una vez tenemos estos posibles valores, es necesario conseguir la suma del total acumulado para cada valor, o, lo que es lo mismo, cuántas veces ha sido repetido un mismo valor entre el dataset para un atributo en concreto. Este total acumu-

lado nos indica la presencia de ese valor dentro de un atributo. Además, este método, como su nombre indica, clasifica el dataset en tantos grupos como valores posibles tenga la clase. Este trabajo es necesario ya que más tarde el total acumulado conseguido anteriormente será dividido entre el total acumulado del valor de la clase al que pertenezca. En conclusión, esta división indicará para cada valor de un mismo atributo, que influencia o peso tiene sobre la clase a la hora de decidir su valor. Las predicciones que genera este clasificador utilizan este porcentaje de influencia de cada valor para cada atributo y el porcentaje de influencia de un valor de la clase entre el total de valores de la clase. En total, nos genera tantos valores como posibles valores tenga la clase, y el máximo entre ellos será la predicción final.

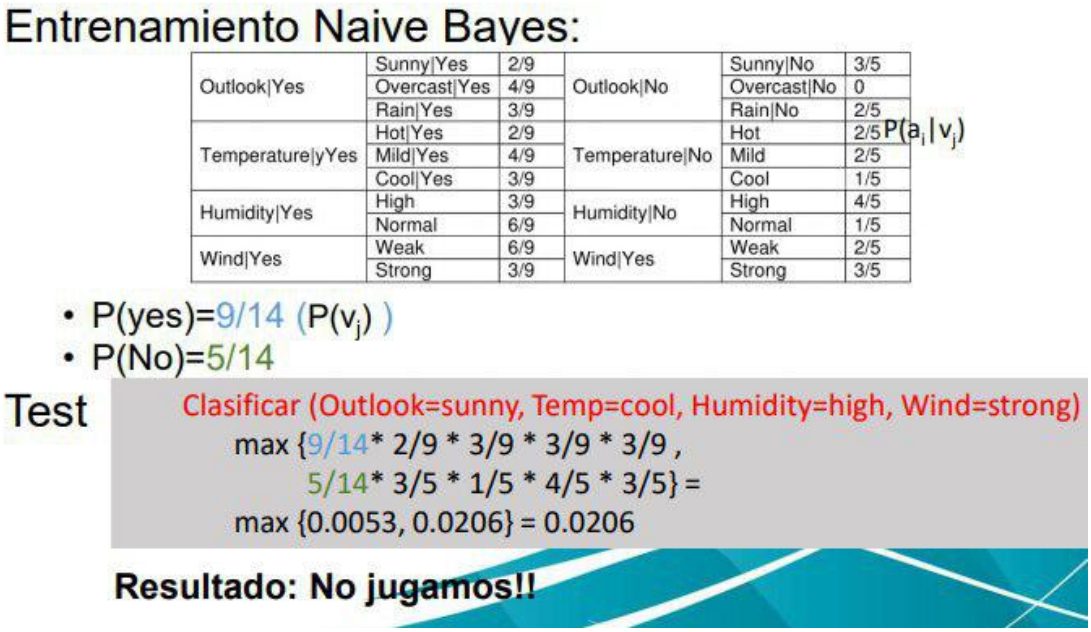


Figure 4.1: Ejemplo de predicción NaiveBayes

Así pues, estos fueron los pasos necesarios para poder aplicar este método a nuestro dataset, el utilizado para generar los modelos del clasificador:

1. Separar el dataset entre los alumnos con alarmas y los que no. Descartar los que no tenían alarma.

2. Calcular para cada valor de cada atributo, cuantas veces se repetía a lo largo de los 240 alumnos. Incluido el valor de la clase, “Alarma” respecto a su otro valor “No Alarma” esto es, el número de alarmas esa semana.

3. Una vez conseguimos esto, realizar la división entre el total acumulado para cada valor de cada atributo entre el total acumulado del valor “Alarma” de la clase.

4. El resultado será, para cada posible valor de un atributo, un peso o porcentaje a la hora de generar la alarma.

5. Repetir este proceso por 10 semanas.

Cabe mencionar que en nuestro caso solamente nos interesaba saber el motivo por el que se había generado el valor “Alarma” de la clase, ya que las predicciones habían sido calculadas automáticamente mediante el script de R.

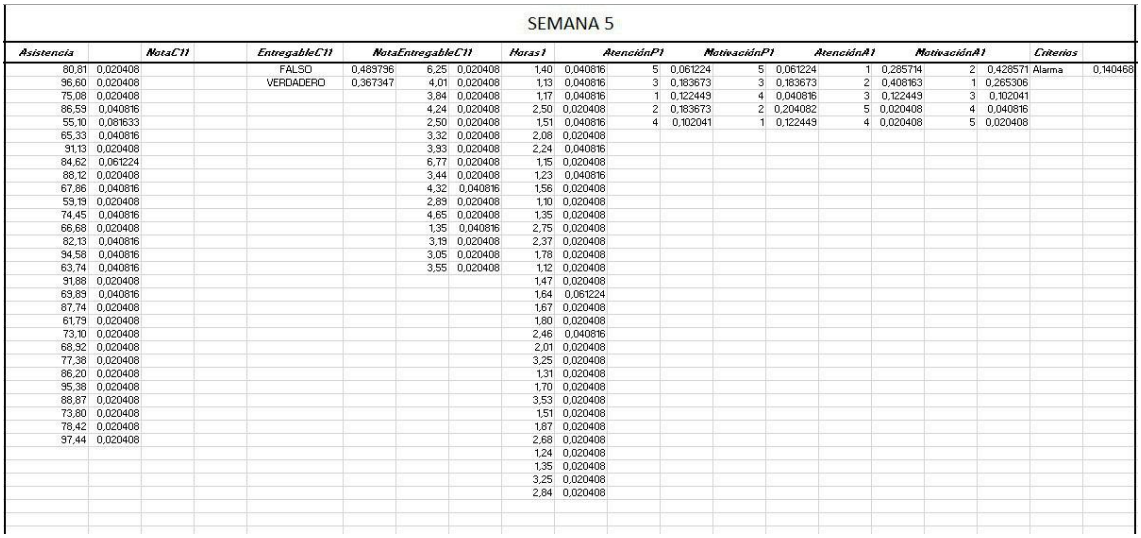
Nos interesaba mantener esta información ajena a cualquier otro dato de este proyecto y, por ello, se generó un fichero por semana al que se accederá mediante el servicio de enseñanza automática que se definirá más adelante.

Así pues, cada vez que un alumno tenga una alarma, se tendrá en cuenta en qué semana se encuentra y se accederá a dicho fichero para saber qué peso tiene cada valor de cada atributo de ese alumno a la hora de generar una alarma esa semana. Además dado que existen ciertos atributos cuyos valores oscilan entre el 0.00 y el 9.99 no se ha podido llegar a contemplar todos y cada unos de los valores, ya que en el dataset inicial solo se recogen muestras de 300 alumnos y entre ellos los valores que llegan a repetirse no cubren todo el abanico de posibles valores para ese atributo.

SEMANA 5

Asistencia		NotaC1		EntregableC1	NotaEntregableC1		Horas1	AtenciónP1		MotivaciónP1		AtenciónA1		MotivaciónA1		Criterios	
80,81	1			FALSO	24	6,25	1	1,40	2	5	3	5	3	1	14	2	21	Alarma	42
96,60	1			VERDADERO	18	4,01	1	1,13	2	3	9	3	9	2	20	1	13		
75,08	1					3,84	1	1,17	2	1	6	4	2	3	6	3	5		
86,59	2					4,24	1	2,50	1	2	9	2	10	5	1	4	2		
55,10	4					2,50	1	1,51	2	4	5	1	6	4	1	5	1		
65,33	2					3,32	1	2,08	1										
91,13	1					3,93	1	2,24	2										
84,62	3					6,77	1	1,15	1										
88,12	1					3,44	1	1,23	2										
67,86	2					4,32	2	1,56	1										
59,19	1					2,89	1	1,10	1										
74,45	2					4,65	1	1,35	1										
66,68	1					1,35	2	2,75	1										
82,13	2					3,19	1	2,37	1										
94,58	2					3,05	1	1,78	1										
63,74	2					3,55	1	1,12	1										
91,88	1							1,47	1										
69,89	2							1,64	3										
87,74	1							1,67	1										
61,79	1							1,80	1										
73,10	1							2,46	2										
68,92	1							2,01	1										
77,38	1							3,25	1										
86,20	1							1,31	1										
95,38	1							1,70	1										
88,87	1							3,53	1										
73,80	1							1,51	1										
78,42	1							1,87	1										
97,44	1							2,68	1										
								1,24	1										
								1,35	1										
								3,25	1										
								2,84	1										

Figure 4.2: Ejemplo de semana con las veces repetidas por cada valor de atributo.



SEMANA 5

Asistencia		NotaC11	EntregableC11	NotaEntregableC11		Horas1		AtenciónP1		MotivaciónP1		AtenciónA1		MotivaciónA1		Criterios	
80,81	0,020408		FALSO	0,483796	6,25	0,020408	1,40	0,040816	5	0,061224	5	0,061224	1	0,285714	2	0,428571	Alarma	0,140468
96,60	0,020408		VERDADERO	0,367347	4,01	0,020408	1,13	0,040816	3	0,183673	3	0,183673	2	0,408163	1	0,265306		
75,08	0,020408				3,84	0,020408	1,17	0,040816	1	0,122449	4	0,040816	3	0,122449	3	0,102041		
86,59	0,040816				4,24	0,020408	2,50	0,020408	2	0,183673	2	0,204082	5	0,020408	4	0,040816		
55,10	0,081633				2,50	0,020408	1,51	0,040816	4	0,102041	1	0,122449	4	0,020408	5	0,020408		
65,33	0,040816				3,32	0,020408	2,08	0,020408										
91,13	0,020408				3,93	0,020408	2,24	0,040816										
84,62	0,061224				6,77	0,020408	1,15	0,020408										
88,12	0,020408				3,44	0,020408	1,23	0,040816										
67,86	0,040816				4,32	0,040816	1,56	0,020408										
59,19	0,020408				2,89	0,020408	1,10	0,020408										
74,45	0,040816				4,65	0,020408	1,35	0,020408										
66,68	0,020408				1,35	0,040816	2,75	0,020408										
82,13	0,040816				3,19	0,020408	2,37	0,020408										
94,58	0,040816				3,05	0,020408	1,78	0,020408										
63,74	0,040816				3,55	0,020408	1,12	0,020408										
91,88	0,020408						1,47	0,020408										
69,89	0,040816						1,64	0,061224										
87,74	0,020408						1,67	0,020408										
61,79	0,020408						1,80	0,020408										
73,10	0,020408						2,46	0,040816										
68,92	0,020408						2,01	0,020408										
77,38	0,020408						3,25	0,020408										
86,20	0,020408						1,31	0,020408										
95,38	0,020408						1,70	0,020408										
88,87	0,020408						3,53	0,020408										
73,80	0,020408						1,51	0,020408										
78,42	0,020408						1,87	0,020408										
97,44	0,020408						2,68	0,020408										
							1,24	0,020408										
							1,35	0,020408										
							3,25	0,020408										
							2,84	0,020408										

Figure 4.3: Ejemplo de semana con los porcentajes por cada valor de atributo (fichero de semana 5).

4.3.3 Conclusiones

Evolución de las alarmas de un alumno: Como hemos mencionado anteriormente el script en R nos indicará que alumnos tiene alarmas y cuáles no, por tanto, a lo largo del año podremos construir un gráfico viendo que semanas han sido más críticas para un alumno en concreto. Evolución de las alarmas de una clase: Al igual que en el apartado anterior podremos apreciar qué asignaturas o qué semanas han sido las peores en cuanto al rendimiento de toda una clase. Evolución de la influencia de cada atributo a la hora de generar alarmas: Con la intención de conocer en cada semana cuáles han sido los atributos que han acarreado alarmas en general y cuales han sido los más importantes. Haciendo uso de las correlaciones, podremos realizar gráficos de la evolución por atributo y/o semana. Evolución de los atributos que han generado una alarma en un alumno: Ya que sabemos los valores de los atributos para cada alumno cada semana, podremos saber qué influencia han tenido estos a la hora de generar una alarma de forma personalizada.

4.3.4 Análisis de los resultados

Ejemplo nuevo alumno en semana 3 con alarma:

Asistencia	Nota	Entregable	NotaEntregable	Horas	AtencionP	MotivacionP	AtencioaA	MotivacionA	Clase
66.68	NA	VERDADERO	1.43	1.42	2	2	3	3	Alarma

Figure 4.4

Resultado del programa teniendo en cuenta el fichero de la semana 3: Atributo:Asistencia Valor:66.68 Peso:0.02173913 Atributo:EntregableC11 Valor:VERDADERO Peso:0.630434783 NO EXISTENTE-Atributo:NotaEntregableC11 Valor:1.43 Peso:0.043478261 Atributo:HorasC11 Valor:1.42 Peso:0.043478261 Atributo:AtencionP1 Valor:2 Peso:0.239130435 Atributo:MotivacionP1 Valor:2 Peso:0.217391304 Atributo:AtencionA1 Valor:3 Peso:0.217391304 Atributo:MotivacionA1 Valor:3 Peso:0.152173913 Atributo:Criterios Valor:Alarma Peso:0.153846154

Porcentaje de influencia respecto a la alarma ordenado por atributo: [EntregableC11=0.63043478 AtencionP1=0.239130435, AtencionA1=0.217391304, MotivacionP1=0.217391304, Criterios=0.153846154, MotivacionA1=0.152173913, NotaEntregableC11=0.043478261, HorasC11=0.043478261, Asistencia=0.02173913]

Aplicamos parte de la fórmula del Clasificador Bayesiano Ingenuo:

0.63 * 0.23 * 0.21 * 0.21 * 0.15 * 0.04 * 0.04 * 0.02 * **0.15**

Como podemos ver, los atributos que más han afectado a que a este alumno tenga un alarma son los siguientes:

1. Entregable
2. Atención por parte del profesor
3. Atención por parte del alumno
4. Motivación por parte del alumno

Conclusión: Los valores de los atributos que son decimales no van a tener una gran influencia en la alarma ya que han sido analizamos únicamente 300 alumnos. Cuantos

más grande fuera la muestra de alumnos, mayores las posibilidades de que estos valores se repitan y, por tanto, la influencia en la alarma sería más exacta. Si bien es cierto que este problema podría solucionarse mediante un sistema de valoración que se basa en nominales, como, “Alto”,”Medio”,”Bajo”...No sería realista ya que hoy en día las universidades valoran con una nota del 0 al 10, decimales incluidos.

Variación de alarmas por semana:

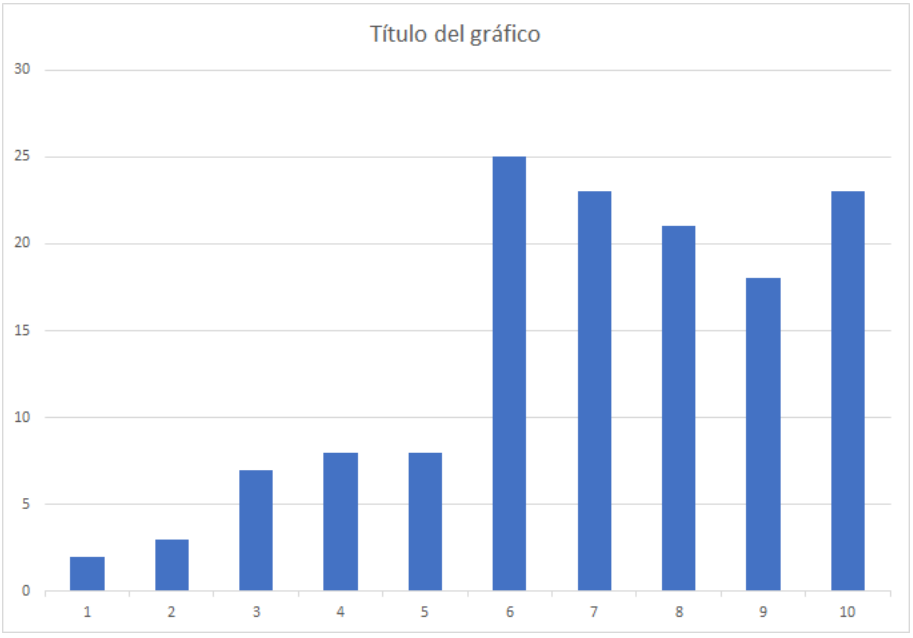


Figure 4.5: Variación de alarmas por semana.

Como podemos apreciar en este gráfico, el número de predicciones por semana aumenta drásticamente a medida que avanza el año. Además el salto más grande se puede observar en la sexta semana donde se han realizado ya los exámenes y por ende, los alumnos que hayan suspendido o hayan sacado una nota más baja de lo habitual, serán avisados a tiempo antes de que la situación empeore.

Conclusión: Como podemos apreciar tanto en la tabla como en los diferentes gráficos de las variaciones de las correlaciones por cada semana(9), algunos atributos como la motivación recibida por el alumno, indica que en las primeras semanas del curso, este atributo tiene una gran influencia para que el valor de la clase sea “Alarma” y que, por

tanto, será un aspecto a tener en cuenta a la hora de dar clase ya que será necesario motivar a los alumnos las primeras semanas del curso ,pero no será tan importante al final.

En general, todos los gráficos indican que a partir de la semana 5, esto es, la mitad del curso transcurrido, la mayoría de los atributos tienen un gran influencia en la alarma. Sin embargo, hay un atributo en particular que tiene valores más altos que los demás, la nota(0.65). Y es que, al fin y al cabo, va a ser la que mayor peso tenga siempre a la hora de decidir si sacar una alarma o no por que será al final el atributo que indique si un alumno pasa o no de curso.Un estudio(9) indica que el coeficiente de correlación Kendall comparado a otros coeficientes como el Pearson o Spearman indica con un 0,7 lo que otros indican como un 0.9 de correlación por lo que la nota está cerca de decidir en algunas semanas directamente si un alumno tiene alarma o no. La asistencia también es un atributo a tener en cuenta, pero que variará dependiendo de la centro en el que se implante el sistema. No todos los centro dan importancia a acudir a las clases lectivas.

Por último, destacar que los valores negativos en el coeficiente de correlación indican que a medida que ese atributo aumenta, el valor de la clase decrece, esto es, está más cerca de llegar a ser una alarma.

Evolución de las correlaciones por semana:

Semana	1	2	3	4	5	6	7	8	9	10
Asistencia	0,21	0,31	0,28	0,29	0,35	0,03	-0,12	0,11	-0,04	-0,12
NotaC11	NA	NA	NA	NA	NA	0,65	0,66	0,68	0,66	0,66
EntregableC11	NA	NA	-0,12	0,18	0,18	0,38	0,00	0,05	0,08	0,00
NotaEntregableC11	NA	NA	0,07	0,25	0,21	0,38	0,08	0,02	0,11	0,08
Horas1	-0,07	0,12	-0,25	-0,01	-0,01	0,00	-0,09	0,10	0,07	-0,09
AtenciónP1	0,03	-0,10	-0,16	-0,04	0,09	0,06	-0,08	0,16	-0,09	-0,08
MotivaciónP	0,01	0,28	-0,02	0,11	-0,11	-0,07	0,10	0,02	-0,13	0,10
AtenciónA	0,17	0,11	0,28	0,00	0,28	-0,16	0,05	0,28	0,16	0,05
MotivaciónA	0,21	0,18	0,37	0,17	0,10	0,02	0,15	0,02	0,12	0,15

Figure 4.6: Correlaciones por semana.

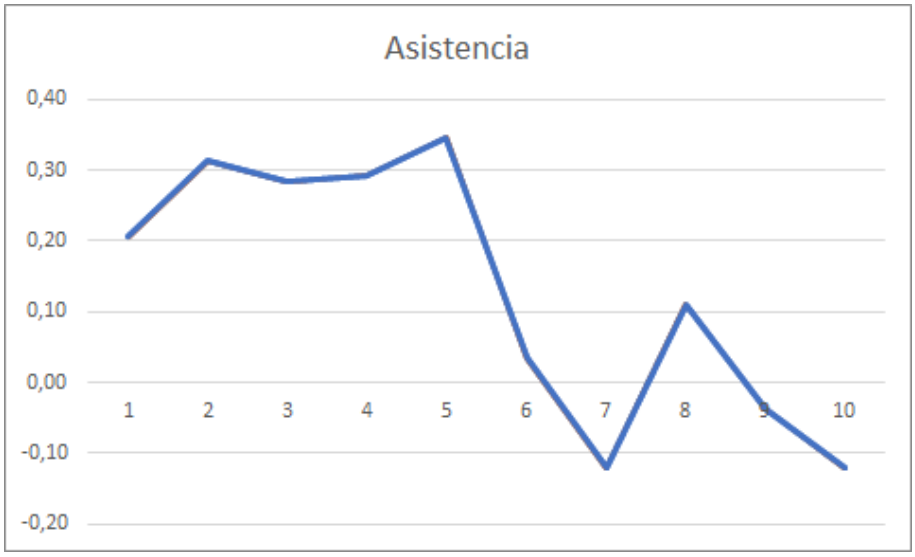


Figure 4.7: Gráfico de la variación del coeficiente de correlación del atributo asistencia entre semanas.

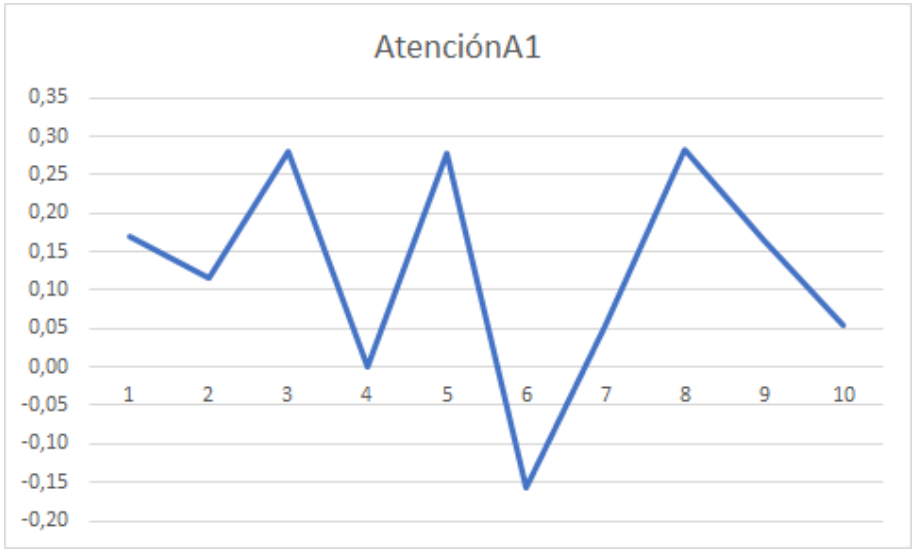


Figure 4.8: Gráfico de la variación del coeficiente de correlación del atributo atención indicado por el alumno entre semanas.

Figure 4.9: Gráfico de la variación del coeficiente de correlación del atributo motivación indicado por el alumno entre semanas.

5. CHAPTER

Infraestructura

5.1 Introdución

Para servir la solución que proponemos se necesita una plataforma que pueda llevar a cabo todo el proceso mencionado en apartados anteriores. Una plataforma robusta, escalable y de alto rendimiento. Para ello la estrategia de Teacheck es separar sus distintos componentes que forman el sistema en microservicios, esto porque Teacheck apuesta por un sistema distribuido. La razón para esta decisión es la solución que queremos dar, para llegar a conseguir lo propuesto se necesitan diversos procesos que tratan de solucionar distintos problemas, y las soluciones proporcionadas por cada servicio conforman la solución final.Así pues, cada microservicio es autónomo y escalable.

Por lo tanto separando estos procesos en pequeños sistemas es lo ideal para Teacheck.

Para ello se decidió dividir el sistema en cinco partes cada una de ellas representado un servicio, que son los siguientes:

- Aplicación Web
- Servicio de acceso a datos
- Servicio de Machine Learning
- Conversor
- Servicio de mensajería

A seguir se explicará la implementación de la infraestructura que proporciona las características de robustez, escalabilidad y resistente.

5.2 Estrategia de la infraestructura

Todos los servicios y la arquitectura del sistema Teacheck está basado en la tecnología de contenerización Docker. Como explicado en apartados anteriores Docker nos permite aislar un entorno virtual del sistema operativo hospedero permitiendo que el entorno sea independiente de plataforma. Además de que podemos crear nuestros propios entornos y desplegarlos en cualquier servidor.

Dicho esto la implementación del sistema Teacheck está pensado para el despliegue en una red de área local, en el dominio de la universidad cliente. Los requerimientos de implementación consiste en tener Docker instalado en una máquina dedicada. Teacheck se despliega utilizando el modo Swarm de docker en dicha máquina, convirtiendo esta última en un nodo gestor del Swam (Cluster).

El Docker Engine instalado en cada máquina viene con un modo que puede ser activado ,dicho modo es Swarm. El objetivo de este modo es controlar de manera nativa un cluster de diversos Docker Engines que pueden estar distribuidos en distintas máquinas. Swarm orquestra y controla todos los servicios desplegados en él así como la comunicación entre nodos y la seguridad de la misma.

El Swarm implementa seguridad por defecto y el intercambio de datos entre todo los nodos es encriptado y políticas de autenticidad también son aplicadas. Aparte de todos estas características Teacheck también eligió Swarm por la facilidad de como se puede escalar cada servicio.Nos permite la creación de réplicas de cada contenedor en cualquier nodo con un cambio de configuración y el redespliegue simplemente actualiza los servicio con la nueva configuración sin necesidad de parar los servicio, es decir casi no tienen downtime.

Docker y el modo Swarm son temas muy extensos y están fuera del alcance de esta sección. En el apartado de manutención y monitorización se llevará a cabo un análisis más detallado de este sistema.

Dicho esto podemos hablar de cómo Teacheck ha utilizado esta tecnología para soportar su solución.

Teacheck dispone de 6 microservicios en total. Cada uno de estos servicios están con-

tenerizados y tienen su propio stack. El stack es simplemente la definición de despliegue de cada sistema, con su configuración, contenedores de soporte y configuración específica para el Swarm (el driver de internet, especificaciones de despliegue como número de réplicas, políticas de redespliegue, políticas de actualización, etc). Los contenedores de soporte le llamamos a load balancers, reverse proxies, etc. Cada servicio, con excepción del Conversor, tiene un load balancer por delante para mantener la carga de los servidores estable. Los load balancers están configurados para utilizar la estrategia de round robin a la hora de repartir la carga ya que existen múltiples réplicas de cada servicio. La configuración específica del Swarm como mencionado define políticas para el despliegue de cada sistema y del stack así como otros varios aspectos que servirán para manutención y actualización. Todos los servicios tiene una política de redespliegue con múltiples intentos y intervalos entre cada uno además de las políticas de actualización de contenedor que está configurado para utilizar paralelismo a la hora de actualizar las diferentes réplicas. Es decir dependiendo de cuántas réplicas de un servicio existen las actualizaciones se pueden aplicar a múltiples réplicas a la vez y no de una en una que es la configuración por defecto.

Por lo tanto Teacheck tiene seis stacks diferentes que se desplegaran en la misma red virtual. Los servicios de cada stack pueden escalar a partir de la modificación de la cantidad de réplicas en la configuración y en seguida la actualización del stack en el Swarm para que se apliquen los cambios. Todo esto se puede llevar a más allá con la creación de nuevos nodos, es decir, Docker Engines que corren en otras máquinas pueden participar del Swarm donde se ubica el sistema de Teacheck. Así se forma el cluster con múltiples nodos y el Swarm tiene la capacidad de balancear los contenedores entre esos diferentes nodos. Esta capacidad viene de que cada gestor en el Swarm asigna a cada servicio un nombre DNS que permite el acceso a cualquier contenedor independiente del nodo en que este. Además del despliegue de nuevos contenedores se puede realizar en cualquier nodo del swarm.

Dicho esto a seguir se explicará cada servicio que compone el sistema Teacheck.

5.3 Servicios

La implementación de cada servicio se basa en el Toolkit Vertx. Este toolkit no es un framework de por sí y se puede utilizar de manera embebida en cualquier aplicación. Vertx permite un flujo de desarrollo asíncrono y está hecha para el desarrollo de microservicios

aunque tiene módulos de soporte para el desarrollo de aplicaciones web. Se eligió Vertx por su alta escalabilidad, optimizando concurrencia en la utilización minimizada de hilos para la gestión de tareas y de alto rendimiento.

Todos los servicios de Teacheck se comunican con uno central que es el servicio de acceso a datos. Por lo tanto se explicará a seguir cada servicio por separado, incluyendo el central, y al final la composición de todos los servicios y el flujo de operación.

5.3.1 Conversor

Teacheck ofrece su solución a distintas instituciones y cada una de ellas ya tienen una manera predeterminada de cómo manipular sus datos y compartirlos con otros servicios. Como dicho en apartados anteriores los datos de los alumnos proporcionados por las universidades son esenciales para lograr nuestro objetivo, por lo que es necesario la obtención de dichos datos en un formato adecuado que nuestro sistema pueda entender y manipular. Teacheck no requiere a sus clientes que se adapten, en este nivel , a su sistema. Para ello ofrecemos el servicio Conversor, que tiene como objetivo servir de puente que traduce los datos que provienen del sistema de la institución a nuestro sistema. En resumen el Conversor recibirá los datos de la institución y los convertirá al formato que nuestro sistema utiliza de manera universal, que es JSON.

Como hemos mencionado, puede que la universidad nos proporcione los datos de los alumnos en formato XML, por lo debemos convertir esos datos en formato JSON. Para ello, seguiremos unos pasos que se explicarán a continuación.

El primer paso es crear un XSD. Un esquema XSD es un mecanismo para comprobar la validez de un documento XML, es decir, definir su estructura: qué elementos, qué tipos de datos, que atributos, en qué orden, cuántas veces se repiten, etc. Con este esquema, podremos validar todos los datos que nos lleguen en formato XML. Así, nos aseguramos que los datos no salen de sus límites establecidos en el XSD. Para la validación, existen diferentes librerías que nos ayudan como java.xml.validation.SchemaFactory, java.xml.validation.Schema, java.xml.validation.Validator, entre otras, pero estas son las que hemos utilizado. Una vez que hemos recibido el XML y lo hemos validado, no podemos pasar de un formato a otro diréctamente. Para ello, vamos a parsear los datos a Objetos de java, es decir, gracias a JAXB, podremos deserializar los XMLs en objetos Java, y, cuando terminemos, solo deberemos convertir esos objetos java en formato JSON. Para esto último, utilizaremos GSON, una librería de código abierto que nos permite la serialización de objetos Java

representandolos en notación JSON.

Para finalizar, solo debemos crear un cliente web y enviar el string en formato JSON al servicio de la base de datos para que él guarde estos datos en la base de datos de Teacheck.

Para asegurarnos de que el cliente web del conversor funciona, se ha decidido hacer un simulador que recibe el JSON que envía el servicio conversor de formato y lo muestra en pantalla. Para ello, simplemente se ha hecho algo parecido que en el servicio anterior. Se ha implementado una API que recibe el JSON y lo muestra en la consola. Con esto, no queremos ver si el conversor a traducido bien el XML a formato JSON, sino comprobar que la comunicación entre los dos servicios se ha efectuado correctamente.

5.3.2 Servicio de machine learning

Para realizar el seguimiento de los alumnos semanalmente se necesita hacer un escaneo de los datos. Estos escaneos ocurren a cada semana y tienen como objetivo monitorizar el rendimiento de cada alumno. Para lograr esto se ha implementado un pequeño sistema que tiene como núcleo un programa de Machine Learning que analiza los datos recibidos y hace una predicción sobre esos datos. Los resultados de este proceso son esenciales para hacer el seguimiento del alumnado y detectar posibles deterioros en su rendimiento.

Este sistema fue diseñado solamente para recibir datos en formato y estructura específica y devolver una nueva estructura de datos con los resultados de predicciones para cada alumno y sus correlaciones. Así pues, el servicio dispone de un REST API compuesto de dos endpoints que servirán tanto para nuestro escaneo semanal como para el testeo.

La estrategia de tener dos endpoints aquí es porque queremos ofrecer un servicio de machine learning que no esté atado solamente a nuestro sistema pero que también se pueda consumir por cualquier otra persona. Por un lado el servicio se comunica con el subsistema central para la ingestión de datos semanalmente con el objetivo de efectuar el escaneo y por otro lado expone un servicio que se puede consumir de manera independiente del resto del sistema.

La ingestión de datos semanalmente puede ser de grande cantidad dependiendo de la institución. Cada alumno tiene una docena de campos a analizar y es costoso para el sistema que recibe esos datos enviarlos al programa núcleo de machine learning y procesar los resultados para que tenga una estructura adecuada a la hora de persistir. El programa de Machine Learning es simplemente un pequeño servidor con un único endpoint que ejecuta el proceso de predicción. Se puede decir que en un principio no haría falta otro programa

por delante, sería ineficiente, pero hablamos de probablemente miles de datos y eso sería una carga muy alta para ese servicio aparte del procesamiento final de los resultados. Por esa razón el servicio de Machine Learning está dividido en dos capas, el servidor principal y el servidor de predicción que llamamos de núcleo. Por lo tanto para balancear esa carga de procesamiento el servidor principal se encarga de separar esos datos en trozos y paralelizar la petición de predicción para cada uno. Claramente no solucionamos el problema de la carga en su totalidad aun tenemos una cantidad decente de peticiones y cada una con miles de datos para procesar. Para solucionar este problema se decidió crear múltiples réplicas del núcleo y por delante del mismo ubicar un balanceador de carga. Este último utiliza la estrategia de round robin para balancear la carga. Entonces de esta manera conseguimos que todo este proceso se agilice y solucionamos el problema de la carga sobre un único servicio. Ahora bien, solamente podemos paralelizar los datos de diferentes asignaturas enviados al servicio, ya que nos interesa analizar la información de la clase en conjunto por las correlaciones que aparecen entre los atributos de los alumnos y la clase que se predice. Por ejemplo, si disponemos de 5 asignaturas y en cada una hay 60 alumnos, podríamos crear 5 hilos diferentes que accederán de forma simultánea al servicio núcleo de predicción. Sin embargo, no nos interesa dividir esos alumnos dentro de una misma asignatura ya que la información que sacaremos de las correlaciones sería diferente.

En definitiva tendríamos el servidor principal que se ocupa de paralelizar todas las peticiones en diversos trozos y procesar los resultados de cada una, seguido de las múltiples instancias del servidor núcleo que se ocuparía de las predicciones. Después de reunir todos los resultados el servicio de ML tendría que persistir en la base de datos mediante el servicio de acceso a datos y su ciclo de operación semanal estaría completo.

Una vez hemos recogido los resultados que el servicio núcleo nos proporcionará y sabemos qué alumnos tienen alarma, se deberá acceder a la información generada gracias a el análisis que se realizó del clasificador bayesiano ingenuo.

Como se ha descrito en el apartado de inteligencia artifical, para saber qué atributos de un alumno en concreto han sido claves a la hora de generar la alarma es necesario acceder a los ficheros que relacionan el valor de cada atributo con un porcentaje de influencia respecto a ese atributo para un cierto valor de la clase, en este caso, “Alarma”. Para ello, y sabiendo a qué semana pertenece la información relativa al alumno, se recogerá la información de dicho fichero y será almacenado. Estos información contiene por un lado, todos los posibles valores de cada atributo y, por otro lado, el porcentaje de influencia dentro de ese atributo con respecto a la clase.

Así pues, una vez leída la información del fichero, se accederá a la información de un alumno en concreto y en base a sus resultados en los diferentes atributos, se relaciona con el porcentaje de influencia previamente almacenado.

De esta forma, y gracias a la información que relaciona cada atributo con su peso y es ordenado de mayor a menor, seremos capaces de identificar cuáles han sido los detonantes más importantes de una alarma en concreto.

El segundo endpoint como dicho antes sirve para el consumo de entidades externas al sistema. El objetivo es el mismo pero la cantidad de datos es menor con lo cual la paralelización no es necesaria. Esta parte del servicio es simplemente para testeo del servicio de predicción por parte otras entidades.

5.3.3 Aplicación web

La aplicación es un componente esencial para la solución final. El objetivo de la aplicación es la monitorización de todos los alumnos así como obtención de datos mediante las encuestas propuestas a los alumnos. Por lo tanto la funcionalidad principal es la visualización de datos.

Este servicio se comunica directamente con el sistema de acceso a datos para manipular los datos necesarios. El frontend de la aplicación se basa en server-side rendering con templates dinámicos. Los templates utilizan estructuras JSON para manipular los datos en la parte del frontend. Para renderizar dichos templates tiene implementado un template engine handlebars.

El servicio tiene todos sus recursos protegidos en el backend mediante autenticación y autorización por roles. La aplicación web tiene tres roles diferentes, profesores, alumnos y coordinadores. Cada una de estas entidades tienen permisos distintos para acceder a recursos y vistas distintas ya que queremos enseñar datos de monitoramento en respecto a cada rol.

Para proteger los recursos de la aplicación el stack para este sistema contiene un servidor que controla la gestión de acceso y de identidad. Este último se llama Keycloak que sirve como el Identity Provider para la aplicación web. Por lo tanto la gestión de acceso a nuestra aplicación lo gestiona el IDP (identity provider). Cuando el usuario se autentica con el proveedor consignará un token que indica la autenticidad del usuario y que tiene libertad para acceder a los recursos según el rol que tenga.

En definitiva la aplicación refleja los resultados del análisis y procesamiento continuo que emplea nuestro sistema.

5.3.4 Servicio de mensajería

Teacheck provee un servicio adicional y que se puede implementar en cada institución, el control de asistencia. Este control de asistencia se haría mediante un dispositivo que sirve de receptor. Cada alumno al entrar en horario de clase debería pasar su tarjeta universitaria en el dispositivo que enseguida envía esos datos al servicio de mensajería de nuestro sistema que su vez persiste los datos recibidos mediante el servicio de acceso a datos.

Este servicio registra la asistencia diaria y datos sobre el alumno (identificador, nombre, etc). La asistencia se da por válida cuando se termina el horario clase normal. El alumno volveria pasar la tarjeta universitaria y el sistema validará la asistencia en respecto a ese dia como válida.

Por lo tanto el objetivo de este servicio es simplemente hacer el registro de asistencia por alumno en nuestro sistema. Esto nos proporcionará más precisión en relación a la asistencia del alumno y mejoras en el sistema de Machine Learning.

5.3.5 Servicio de acceso a datos

Todos los datos manejados por el sistema está ubicado en una base de datos central que sólo es accesible a través del servicio de acceso de datos. Este servicio es el responsable de todas las transacciones a la base de datos. Dispone de una API compuesta para todos los servicios del sistema. Si algún otro servicio necesita datos tiene que pasar por este sistema.

La API de este servicio está dividida en 4 partes. Cada parte está relacionada con un servicio del sistema, aplicación web, conversor, sistema de mensajería y servicio de machine learning. El sistema es totalmente asíncrono y reactivo. Ninguna acción en el sistema es bloqueante, con excepción de las migraciones iniciales a la base de datos, lo que permite que el sistema esté activo, capaz de atender y procesar todas las peticiones de los demás servicios con lo mínimo de retardo. El objetivo de este microservicio es procesar todas las transacciones y consultas que los demás servicios requieren para lograr sus propios objetivos, es un componente crucial para el funcionamiento de Teacheck.

5.4 Seguridad

5.4.1 Introdución

Teacheck es un sistema distribuido donde en todo momento se intercambia información sensible de un servicio a otro. La seguridad en este sistema es imprescindible para asegurar nuestros servicios de amenazas. Por lo tanto es importante identificar los diversos factores que podrían tener impacto negativo a nuestro sistema y intentar mitigarlos de alguna manera. Para ello se decidió hacer un análisis de riesgos donde analizaremos nuestro sistema en busca de las posibles amenazas a las cuales nuestro sistema está expuesto.

5.4.2 Análisis de riesgos

ALCANCE

Este análisis de riesgos trata de centrarse en el proceso más importante de la empresa, en nuestro caso, el comercial. La venta de los servicios de Teacheck es lo que trae beneficio y lo que mantiene el negocio activo, por lo tanto, es de gran importancia que el alcance sea a nivel de los activos más importantes que mantienen el proceso comercial en la empresa.

ACTIVOS

La principal actividad beneficiaria para Teacheck es nuestra aplicación web. Aparte de esta última existe un sistema aún más complejo que mantiene el flujo de información actualizado y la estabilidad de la aplicación. A continuación se muestra la tabla de activos correspondientes al alcance tomado para el análisis:

ID	Nombre	Descripción	Responsable	Tipo	Ubicación	Crítico
1	Teacheck - Web App	Aplicación web de la empresa	Desarrolladores	Aplicación	No tiene ubicación física	Si
2	Servidor - 01	Servidor de la base de datos.	Administrador de Servidores	Servidor en red local	No tiene ubicación física	Si
3	Servidor - 02	Servidor del servicio de IA	Administrador de Servidores	Servidor en red local	No tiene ubicación física	No
4	Servidor - 03	Servidor conversión de datos	Administrador de Servidores	Servidor en red local	No tiene ubicación física	No
5	Base de Datos - 01	Instancia de la Base de datos de la aplicación web	Gestor de la Base de Datos	DB en la red local	No tiene ubicación física	Si
6	Base de datos - 02	Instancia de la Base de Keycloak IDP	Gestor de la Base de Datos	DB en la red local	No tiene ubicación física	Si
7	Desarrolladores	Empleados de la empresa.	-	Empleados	Departamento de Desarrollo	No

Figure 5.1: Tabla de activos

Arriba, aparte de los diversos servidores que componen nuestro sistema, las personas que lo mantienen también toman gran parte a la hora de analizar los puntos débiles del sistema. Ya que Teacheck es una empresa pequeña, la mayor parte de sus empleados son los mismos desarrolladores.

IDENTIFICACIÓN DE AMENAZAS

Aplicación web

La aplicación web está sujeta a diversos tipos de amenazas que podrían arruinar el negocio de empresa. Pequeños detalles de implementación que pueden ser catastróficos una vez el artefacto entre en producción. Internet se ha vuelto un sitio donde la seguridad es algo indispensable si se desea mantener un negocio en marcha sin desfortunios o mantener privada la propia vida personal. Teacheck es una aplicación que trata con datos sensibles, datos de alumnos y profesores, y este es el punto principal de todo el negocio. Estos datos son esenciales para el cumplimiento de nuestros objetivos como organización. Así pues, esta primera sección se enfocará en la identificación de amenazas relacionadas con la **confidencialidad** de estos datos.

Los ataques de tipo SQL injection son bastante comunes en aplicaciones web. La potencial inyección de una query mal formada a la API de la aplicación puede tener resultados desastrosos, como cuenta la bien conocida historia de Little Bobby Tables. Aquí, una entrada del usuario es construida para convertir un SQL statement en más de uno. Esto consiste en la implantación del terrorífico DROP TABLES en una query. Como se podría preveer, esto trae consecuencias inmensurables.

La mala gestión de sesiones e incluso la autenticación de usuarios en la aplicación pueden acarrear diversos problemas. Para empezar los sessionID, utilizados en la parte del frontend de la aplicación pueden proporcionar problemas relacionados con la sesión del usuario. La mala gestión de las sesiones pueden causar suplantación de identidad y fraude. Si los identificadores de las sesiones son muy visibles o predecibles puede que ocurra un session hijacking, donde un atacante puede tomar el control de la sesión del usuario. Y ,a día de hoy, aún teniendo el id de la sesión embebido en una cookie, esto todavía es posible. Otro riesgo a analizar es la fijación de sesión. Un atacante puede entrar en nuestra aplicación, conseguir un identificador de sesión y pasarlo a un usuario que sin saberlo accede y se autentica en la aplicación utilizando el sessionID del atacante. Así pues, permitiendo al intruso conseguir el control sobre una cuenta activa del sistema.

Las credenciales de los usuarios es uno de los puntos más importantes en la seguridad de nuestra aplicación. Uno de las amenazas es el robo de credenciales. Esto puede ocurrir de varias maneras pero uno de los principales motivos es la manera como se comunica el frontend con el backend. El envío de credenciales en texto plano es uno de los problemas que pueden llegar a ocurrir.

La autenticación de los usuarios también es un factor importante ya que, una vez más, lidiamos con datos sensibles. Existen diversos problemas relacionados con esto, como, por ejemplo, atacantes que intentan autenticarse diversas veces en función de averiguar credenciales o algún tipo de error que ayude a entrar en el sistema de forma no autorizada.

Otro problema muy recurrente en aplicaciones web son los XSS (Cross-site-scripting). Este tipo de amenaza consiste en la ejecución de código JavaScript o otro lenguaje del mismo tipo en las páginas visitadas por un usuario. Normalmente, se utiliza ese tipo de ataque para tomar el control de sesión de un usuario.

La utilización de database IDs en URLs puede ser un problema. Un atacante puede utilizar URLs con parámetros para hacer diversas pruebas contra el backend de la aplicación con el objetivo de averiguar entidades de la base de datos, por ejemplo un usuario. Este tipo de problema ocurre cuando nuestro servicio confunde "tener una URL" con "permitido acceder a un recurso". Los que llaman al servicio pueden tener muchas direcciones URL de rastreo, phishing o sondeo que no deberían tener el permiso para acceder a esos recursos. Si un recurso debe ser enviado a usuarios autorizados, el servicio tiene que comprobarlo en cada llamada. Aquí también puede ocurrir problemas como la fuga de información. Por ejemplo nuestro servicio responde a una llamada con "404, not found" cuando alguien pide un recurso que no existe, pero responde con un "403 not authorized" cuando el recurso existe pero el que llama no tiene permisos, esto es fuga de información.Con esta información un atacante puede deducir que recursos existen y cuales pueden ser accedidos permitiéndole hacer lo que se ha mencionado anteriormente, el sondeo.

Servidores

La seguridad de los servidores es muy importante ya que usualmente contienen una gran cantidad de información vital para la organización. Si un servidor está comprometido, toda la información será vulnerable a un ataque y los siguiente puntos detallan cuales son los mayores riesgos a tener en cuenta.

El primer riesgo lo encontramos a la hora instalar el sistema operativo sin prestar mucha atención a todo lo que se pueda llegar a instalar. Esto puede causar que se instalen servicios por defecto que no necesitamos o deseamos, y en algunas ocasiones, que estos estén activados por defecto también. Estos pueden generar tráfico indeseado y, además de eso, pueden abrir puertos a distintas personas indeseadas con intenciones malignas. Para evitar esto, es de vital importancia cerrar puertos y desactivar servicios inutilizados.

La mayoría de las aplicaciones de servidores cuentan con años de experiencia en el ámbito laboral y esto les ha expuesto a incontables ataques, por lo cual, su código ha sido reforzado y se han ido corrigiendo y reparando tanto pequeñas fallas como grandes errores. Sin embargo, ningún software es perfecto, y menos cuando se trata de algo recién salido al mercado o no cuenta con la suficiente experiencia por falta de popularidad. En muchas ocasiones, tanto desarrolladores como administradores encuentran pequeños bugs en las aplicaciones y las publican en distintas páginas web destinadas a dar a conocer

dichas fallas, para que de esta manera toda la comunidad esté al tanto y pueda darles una solución. Pero todo el mundo, incluido cualquier hacker, puede tener acceso a estas páginas web por lo cual están al tanto de las medidas que se toman y esto le puede llegar a dar pistas de qué pasos tomar a la hora de preparar el siguiente ataque.

Una de las grandes amenazas y mayores causas de la vulnerabilidad de seguridad de los sistemas es dejarla en manos de personas poco entrenadas o no aptas para este trabajo. Esto se aplica a trabajadores nuevos o a aquellos con la insuficiente motivación o formación. Hay diferentes errores que un descuidado administrador puede cometer, y, uno de los más graves es ,por ejemplo, dejar sin modificar las llaves o contraseñas de los servicios. En muchas ocasiones después de la instalación de una base de datos, el desarrollador que lo ha instalado deja la contraseña por defecto pensando que el administrador lo cambiará, algo que no siempre ocurre, dejando una importante vulnerabilidad abierta.

Otro de los mayores riesgos para un servidor ocurre cuando este se despliega en internet y la gran seguridad con la que antes contaba deja de ser tan eficaz. Pero nuestro sistema se despliega localmente lo cual nos ahorra este tipo de problemas.

Por último, la comunicación entre servicios tiene que ser segura ya que es importante que ningún tercero pueda interceptar un mensaje y leerlo. Al estar desplegado localmente no es un alto riesgo para nuestro sistema, pero la manera de realizar las comunicaciones es algo a tener en cuenta.

Bases de datos

La base de datos es uno de los componentes más importantes de nuestro sistema. El sistema de Teacheck esta compuesto por dos bases de datos, una para toda la información de la aplicación y la otra para el servicio de IDP Keycloak. Por lo tanto, es importante analizar las posibles amenazas.

Uno de los problemas más básicos y que ocurre de a menudo, es la configuración errónea de estos sistemas. La configuración por defecto, contraseñas por defecto (admin / admin), etc. Atacantes podrían entrar en el sistema fácilmente y tomar el control.

Los privilegios a cada individuo que tiene acceso autorizado a la base de datos tienen que ser controlados, con el fin de evitar dar más privilegios de los necesarios a las personas o programas que actúan sobre la base de datos. Ofrecer a un usuario privilegios que no son necesarios para sus acciones diarias es como abrir la puerta para un ataque eventual.

Las funcionalidades habilitadas innecesariamente son un problema. El sistema de la base de datos solo debe tener lo esencial para su funcionamiento y para lo que se quiere realizar a diario. Extensiones, dependencias y otros componentes no utilizados pueden causar problemas si tienen alguna vulnerabilidad debido a parches o incompatibilidades. También hay que tener en cuenta los desbordamientos de buffer. Permitir entradas que exceden la capacidad predeterminada de estos buffers pueden causar problemas.

Las actualizaciones de la base datos son importantes y la falta de esas actualizaciones tiene sus riesgos. Estas actualizaciones pueden corregir un bug, mejorar políticas de seguridad, vulnerabilidades y etc. Por lo tanto, es de extrema importancia mantener la base de datos actualizada, eso sí, como se ha comentado anteriormente, siempre analizando los posibles problemas que estas actualizaciones pueden acarrear.

Desarrolladores

Como último punto analizaremos el impacto negativo que el factor humano pueda llegar a tener. Y es que a veces muchas empresas se preocupan en conseguir la mejor tecnología del mercado y se olvidan que muchos de los problemas los pueden evitar y a su vez propiciar los mismos trabajadores.

Y ya se ha mencionado en puntos anteriores que las acciones de los empleados pueden repercutir severamente en la seguridad del sistema como por ejemplo a la hora de poner las contraseñas o a la hora de llevar a cabo el mantenimiento de los servidores, lo que demuestra que el factor humano está implicado en gran número en los ciberataques.

Tabla - Resumen Amenazas

En la siguiente tabla recogemos todas las amenazas a modo de resumen asignándoles un identificador y indicando a cual de los activos afecta cada uno.

ID	Descripción amenaza	ID activo afectado
1	Inyecciones SQL	1
2	Session Hijacking	1
3	Acceso no autorizado a recursos	1
4	XSS	1
5	Ataque desde puerto abierto no deseado	2 & 3 & 4
6	Vulnerabilidad debido a un software o servicio sin sus parches	2 & 3 & 4
7	Administración desatendida	2 & 3 & 4
8	Terceros interceptan información entre servicios	2 & 3 & 4

Figure 5.2: Tabla - Resumen Amenazas 1

9	Configuración por defecto	5 & 6
10	Privilegios mal asignados	5 & 6
11	Funciones innecesariamente habilitadas	5 & 6
12	Actualizaciones no realizadas	5 & 6
13	El factor humano puede causar cualquier tipo error o falla. Con las medidas tomadas tendrá una baja probabilidad de ocurrir pero no podemos especificar el impacto o riesgo que este pueda tener	7

Figure 5.3: Tabla - Resumen Amenazas 2

EVALUAR EL RIESGO

Una vez analizados los riesgos y las medidas a tomar en cuenta, es necesario calcular el riesgo que cada una de esas amenazas supone para nuestro sistema. Para ello, estimaremos la probabilidad y el impacto de los riesgos.

Tabla para el cálculo de la probabilidad

Cualitativo	Cuantitativo	Descripción
Baja	1	La amenaza se materializa a lo sumo una vez cada año
Media	2	La amenaza se materializa a lo sumo cada mes
Alta	3	La amenaza se materializa a lo sumo una vez cada semana

Figure 5.4: Tabla para el cálculo de la probabilidad

Tabla para el cálculo del impacto

Cualitativo	Cuantitativo	Descripción
Bajo	1	El daño derivado no tiene consecuencias relevantes
Medio	2	El daño derivado tiene consecuencias reseñables
Alto	3	El daño derivado tiene consecuencias graves

Figure 5.5: Tabla para el cálculo del impacto

Para el cálculo del riesgo, hemos optado por hacer un análisis cualitativo, y para ello haremos uso de una matriz de riesgo.

Matriz de riesgo

	Impacto bajo	Impacto medio	Impacto alto
Probabilidad baja	Muy bajo	Bajo	Medio
Probabilidad media	Bajo	Medio	Alto
Probabilidad alta	Medio	Alto	Muy alto

Figure 5.6: Matriz de riesgo

Aplicación web

Amenaza	Probabilidad	Impacto	Riesgo
Inyecciones SQL	Medio	Alto	Alto
Session Hijacking	Medio	Medio	Bajo
Acceso no autorizado a recursos	Alto	Alto	Alto
XSS	Baja	Medio	Bajo

Figure 5.7: Aplicación web

Servidores

Amenaza	Probabilidad	Impacto	Riesgo
Ataque desde puerto abierto no deseado	Medio	Alto	Alto
Vulnerabilidad debido a un software o servicio sin sus parches	Medio	Medio	Medio
Administración desatendida	Alto	Medio	Alto
Terceros interceptan información entre servicios	Medio	Bajo	Bajo

Figure 5.8: Servidores

Bases de datos

Amenaza	Probabilidad	Impacto	Riesgo
Configuración por defecto	Bajo	Alto	Medio
Privilegios mal asignados	Medio	Alto	Alto
Funciones innecesariamente habilitadas	Medio	Medio	Medio
Actualizaciones no realizadas	Alto	Medio	Alto

Figure 5.9: Bases de datos

Desarrolladores

Amenaza	Probabilidad	Impacto	Riesgo
El factor humano puede causar cualquier tipo error o falla. Con las medidas tomadas tendrá una baja probabilidad de ocurrir pero no podemos especificar el impacto o riesgo que este pueda tener	Medio	Desconocido	Desconocido

Figure 5.10: Desarrolladores

SALVAGUARDIAS

Aplicación web

Una vez hemos definido las posibles amenazas de nuestra aplicación, es hora de analizar qué podemos hacer para mitigar estas situaciones.

Hemos explicado que las inyecciones SQL eran algo común y que, hoy en día, no hay excusas para no prevenirlo con tantas librerías y buenas prácticas. Para ello, nuestro sistema principal responsable de las transacciones con la base de datos hará consultas con preparedStatments (utilizando marcadores de posición) en el caso de consultas con parámetros. De esta manera evitaremos problemas con cadenas de caracteres SQL malformadas asegurando las consultas y transacciones. Además, se debe comprobar las entradas a nuestra API porque es preferible detectar el problema en los boundaries de nuestra aplicación. Así pues, evitando que el error ocurra en alguna capa más profunda del sistema.

La gestión de sesiones debe seguir algunos criterios:

- Usar una ID de sesión larga con mucha entropía.
- Generar ID de sesión usando un generador de números pseudoaleatorios (PRNG) con buenas propiedades criptográficas.

- Cuando un usuario se autentica, hay que generar una nueva ID de sesión. De esa manera, si un ataque de fijación de sesión ocurre, el atacante no tendrá acceso a la cuenta de usuario.
- Utilizar cookies para intercambiar identificadores de sesión. No se debe aceptar ID de sesión a través de otros mecanismos, algunos servidores emitirán identificadores de sesión en las cookies pero seguirán aceptando a través de los parámetros de consulta. Deshabilitar esta última opción.

Todo tipo de datos sensibles no pueden circular en la red en texto plano. La comunicación entre el frontend y el backend debe estar encriptada. Los certificados SSL proveen este tipo de protección encriptando toda la comunicación entre cliente y servidor. La aplicación tiene por delante un reverse proxy, Nginx, los certificados deben de ser configurados para el mismo con el objetivo de redireccionar tráfico utilizando protocolo HTTP a HTTPS. Los certificados pueden ser fácilmente obtenidos con Let's Encrypt, una autoridad conocida y que provee certificados de forma gratuita.

La autenticación y autorización de los usuario se hará mediante un Identity Provider. El IDP elegido es Keycloak, un sistema que realiza la gestión de acceso e identidad. Este llevará a cabo la autenticación de usuarios así como la autorización mediante roles. Dicho esto, Teacheck tiene tres roles distintos: alumno, profesor y coordinador. El flujo de autenticación se hará en base al protocolo OIDC, una extensión de OAuth 2.0. El protocolo OIDC utiliza estándares de Json Web Tokens. Esos estándares definen un Identity Token en formato JSON y maneras de cómo firmar digitalmente y encriptar esos datos. Existen dos casos de uso cuando hablamos de OIDC, nos centraremos en el primero que consiste en una petición al servidor de Keycloak para que autentique un usuario. Entonces, si el proceso de autenticación se efectúa correctamente el servidor enviará de vuelta dos tokens, Identity Token y un Access Token. El Identity Token contiene información sobre el usuario como email, nombre, etc. El Access Token es firmado por lo que Keycloak llama de realm (lo que gestiona una serie de usuarios, grupos y roles) y contiene información, como los roles, que la aplicación puede utilizar para determinar si un usuario puede acceder a un recurso o no. Aparte de acceso a recursos cada llamada a los endpoints tendrán que añadir una cabecera con su Identity Token, "Authorization Bearer <token>", para que se pueda identificar el usuario y darle acceso a nuestra API.

Para evitar los XSS, como dicho anteriormente, se debe validar todas las entradas que llegan al API de la aplicación. También evitar la construcción de estructura de datos (html por ejemplo) mediante la concatenación de strings. Además existen herramientas que pueden ayudar mitigar este problema como OWASP Java Encoder. Esta librería codifica las estructuras de datos antes mencionadas. Aunque la aplicación utiliza Template Engines, que ya proporcionan su propia codificación de las estructuras HTML puede que se utilice estructura planas personalizadas en el servidor.

Para evitar el acceso directo a objetos, se reducirá el valor del sondeo de URL y se comprobará la autorización a esos objetos.

Servidores

Una vez vistas las amenazas con las que nuestro sistema de servidores tendría que lidiar, analizaremos las medidas tomadas.

Primero debemos asegurarnos que no se instala ningún servicio que no queremos que se instale y que no queden abiertos puertos que no deben. También, a la hora de decidir que software se utilizará, se tendrá en cuenta la antigüedad y experiencia del mismo, para así asegurarnos de que ha sido utilizado por otros y tiene cierto nivel de seguridad.

Otro gran problema como antes hemos mencionado, es tener a cargo del mantenimiento de los servidores y de su seguridad a personas no aptas para ello. Por lo tanto, será importante mantener una formación correcta del personal, para así poder mantener la seguridad de los distintos servidores. Además de una formación inicial a un posible nuevo trabajador, habrá que seguir realizando formaciones periódicamente para así además de mantener al personal formado, también se le mantendrá motivado.

Por último, para poder realizar una comunicación segura entre distintos servicios, empezar hacemos uso de docker swarm, que más tarde se explica, y que básicamente encripta la comunicación entre los distintos contenedores del entorno.

Bases de datos

Como el resto de todos los componentes del sistema las bases de datos estarán contenerizadas y vienen con la configuración por defecto. Toda esa configuración se puede sobrescribir a partir de ficheros de configuración pasados al contenedor. Claramente esos ficheros de configuración son datos no sensibles, como puertos, hostname, nombre de la base de datos , etc. Para datos sensibles hay que utilizar la funcionalidad del entorno en el cual se despliega todos los servicios, Docker swarm. El modo swarm de Docker permite la gestión centralizada de secrets (password, certificados TLS, claves SSH, etc). Con el swarm podemos garantizar que container tiene acceso a qué secrets y en el intercambio son encriptadas así como cuando están guardadas en el swarm. Por lo tanto toda configuración sensible debe utilizar la funcionalidad por defecto de Docker Swarm y la configuración no sensible por variables de entorno o ficheros de configuración.

Se debe evitar los privilegios innecesarios. Todos los usuarios tienen que disponer solamente de privilegios necesarios para efectuar sus tareas. La clasificación de estos privilegios deben de ser registrados en el perfil de cada usuario teniendo en cuenta la frecuencia de operación de cada tarea. Una vez algún privilegio se vuelva innecesario para un usuario se debe retirar dicho privilegio.

En relación a dependencias de las bases de datos es importante mantener solamente lo necesario. Funcionalidades que no se utilizan deben ser desinstaladas.

La actualización de las bases de datos en el caso de un entorno en Swarm consiste en la actualización de la imagen del contenedor. Cómo utilizamos las imágenes oficiales de PostgreSQL para cada base de datos, cada una son mantenidas y actualizadas periódicamente por el desarrollador oficial. Para actualizar hay que tener en cuenta las vulnerabilidades de las imágenes. Estas vulnerabilidades se pueden ver en el repositorio oficial de PostgreSQL en Docker Hub, analizar y validar la posible actualización de la imagen del contenedor en el Swam.

Desarrolladores

En cuanto al personal de la empresa, hemos visto que gran parte de los ciberataques se producen debido a una falta de responsabilidad por parte de los mismos. Para evitar este tipo de situaciones, se tomarán dos medidas al respecto, la primera, informar y concienciar a los trabajadores de la importancia de su responsabilidad y toma de decisiones para la seguridad de la empresa.Y la segunda, tratar de establecer medidas predeterminadas y

hacer uso de la inteligencia artificial para que de esta manera, se libre en la medida de lo posible al trabajador de tomar decisiones en cuanto a la seguridad.

Por ejemplo, se hará uso de una aplicación para monitorizar contraseñas, haciendo de esta manera que se actualicen solas.

TRATAR EL RIESGO

A la hora de tratar los riesgos, se dará prioridad a aquellos más graves, pero se intentará dar solución a todos. La táctica a seguir será implantar medidas de seguridad para dar solución a los riesgos que puedan aparecer. Se ha decidido tomar este camino debido a que la aplicación se instala localmente, no tiene un uso continuo y se conoce cuando se está usando. Por lo tanto, se realizará una copia semanal de la base de datos para así no perder ningún dato y el sistema se apagará en caso de que haya algún riesgo y teniendo en cuenta un par de criterios. El primero de esos criterios es ver si el riesgo impide el correcto funcionamiento del sistema, y el segundo si el riesgo es "medio" o superior.

A su vez, será importante una correcta monitorización de los riesgos, ya que se pueden repetir a lo largo del tiempo.

5.4.3 Análisis de riesgos residual

Una vez realizado el analizadas las amenazas y sus respectivos salvaguardas, pasaremos a analizar las posibles nuevas amenazas generadas a la hora de implementar los salvaguardas. También cabe la posibilidad de que alguna de las amenazas siga sin estar solucionada o controlada del todo.

IDENTIFICACIÓN DE AMENAZAS

Aplicación web

Comenzando con la aplicación web hay que tener en cuenta que la actual librería utilizada para desarrollar la API dispone de un algoritmo para generar ID de sesión random que puede quedarse obsoleto. Es poco probable pero es algo que se debe tener en cuenta si la plataforma pretende estar activa durante mucho tiempo. Otro factor importante para tener en cuenta son las vulnerabilidades pueden aparecer con nuevas actualizaciones de la librería.

La implantación de los certificados para los servidores pueden tener serios problemas en respecto a la caducidad de los mismos. Estos certificados tiene una fecha de caducidad, por lo cual si no se tiene un sistema automático que controle dicha fecha de caducidad, podría pasar la fecha y dejar nuestros servidores comunicándose de manera insegura por la red. Además es impráctico hacer la renovación manualmente, aunque en casos de que un certificado se caduque y lleguemos a tiempo para renovarlo esto requiere un esfuerzo y tiempo que son críticos ya tratamos con datos sensibles en la mayor parte de nuestro sistema.

Por último nuestro IDP Key Cloak es un software de código abierto y tiene su repositorio bastante activo y actualizaciones bastante corrientes. Por lo que aquí puede aparecer un problema como vulnerabilidades en el sistema de Keycloak. Dichas vulnerabilidades pueden ser utilizadas por otras entidades para intentar atacar el sistema. sistema.

Bases de datos

La asignación de los privilegios es importante para el correcto funcionamiento del sistema, y a su vez para que el sistema sea seguro. Dicho esto, en el anterior análisis se especificaba que era importante no asignar privilegios a quien no los necesitaba y que debido a eso se regularía el entregar privilegios. Pero algo que no se contempló fue el no darle a alguien o algo el privilegio que necesita para desempeñar su tarea, ya que si por ejemplo alguna funcionalidad o servicio necesita algún privilegio para su correcto funcionamiento y se le deniega dicho privilegio, el sistema entero podría dejar de funcionar.

Otro de los problemas era el no llevar las actualizaciones al día, y por ello decidimos automatizar el proceso. Esto crea otra amenaza ya que las actualizaciones pueden traer consigo posibles vulnerabilidades que podrían amenazar a nuestro sistema.

Desarrolladores

Tal y como se ha mencionado en el análisis de riesgos anterior, el factor humano es uno de las posibles amenazas de sistema, y aunque se intente dar una solución formando a los trabajadores, o sustituyendo las decisiones de los mismos, mientras haya personas

trabajando, el factor humano seguirá existiendo. Por lo tanto, se puede disminuir la probabilidad del riesgo, pero no hacerlo desaparecer.

EVALUAR EL RIESGO

Una vez vistas las salvaguardas y las posibles amenazas restantes, tanto nuevas como las que no se han conseguido mitigar del todo, volvemos a analizar el riesgo de las mismas.

Aplicación web

Amenaza	Probabilidad	Impacto	Riesgo
Inyecciones SQL	Baja	Alto	Medio
Session Hijacking	Baja	Medio	Bajo
Acceso no autorizado a recursos	Baja	Alto	Medio
XSS	Baja	Medio	Bajo
Desencriptar la comunicación frontend-backend	Baja	Alto	Medio
Caducidad certificados	Baja	Alto	Medio
Keycloak hackeado	Baja	Alto	Medio

Figure 5.11: Aplicación web

Servidores

Amenaza	Probabilidad	Impacto	Riesgo
Ataque desde puerto abierto no deseado	Baja	Alto	Medio
Vulnerabilidad debido a un software o servicio sin sus parches	Baja	Medio	Bajo
Administración desatendida	Media	Medio	Medio
Terceros interceptan información entre servicios	Baja	Bajo	Muy bajo

Figure 5.12: Servidores

Bases de datos

Amenaza	Probabilidad	Impacto	Riesgo
Configuración por defecto	Bajo	Alto	Medio
Privilegios mal asignados	Medio	Alto	Alto
Funciones innecesariamente habilitadas	Baja	Medio	Bajo
Actualizaciones no realizadas	Baja	Medio	Bajo
Denegación incorrecta de privilegios	Baja	Alto	Medio
Actualización con vulnerabilidad	Baja	Medio	Bajo

Figure 5.13: Bases de datos

Desarrolladores

Amenaza	Probabilidad	Impacto	Riesgo
El factor humano puede causar cualquier tipo error o falla. Con las medidas tomadas tendrá una baja probabilidad de ocurrir pero no podemos especificar el impacto o riesgo que este pueda tener	Baja	Desconocido	Desconocido

Figure 5.14: Desarrolladores

SALVAGUARDIAS

Aplicación web

De las nuevas amenazas solo podemos solucionar el relativo a los certificados, automatizando las actualizaciones de dichos certificados.

Servidores

Respecto a la base de datos, viendo que la asignación y control de los privilegios es importante, se le da la importancia y atención que se merece.

6. CHAPTER

conclusión

7. CHAPTER

Futuro

8. CHAPTER

Anexo