Ingenieria Informática

3º curso

Teacheck - Documentación análisis del sistema y organización del proyecto

Autor

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

07-05-2019

Contents

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 Objetivos

Teacheck es una aplicación para realizar el seguimiento del alumnado de forma automática y precisa. El principal objetivo de esta, es reducir en gran parte la tasa de abandono, repetidores y suspensos para así aumentar la tasa de rendimiento y graduación de una universidad. También se van a poder analizar diferentes problemas que puedan surgir a nivel de toda una clase gracias a los diferentes datos que almacenaremos. Con todo esto, conseguiremos aumentar el rendimiento y eficiencia 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. Al tratarse de una aplicación para universidades, estas serán las que compren nuestros servicios tanto para un curso como para todo un grado. Por tanto, todo beneficio económico será el relacionado con nuevas matrículas de los alumnos, que, al fin y al cabo, se han previsto que consigan gracias al prestigio que pueda otorgar la aplicación. Por otro lado, Teacheck se encargará de conseguir el formato necesario para alimentar el sistema, así pues, la universidad únicamente deberá dar acceso a la fuente de dichos datos, minimizando así los costes de implementación de la aplicación por parte de las universidades.

2.2 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.

2.2.1 Organigrama de funcionamiento

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.

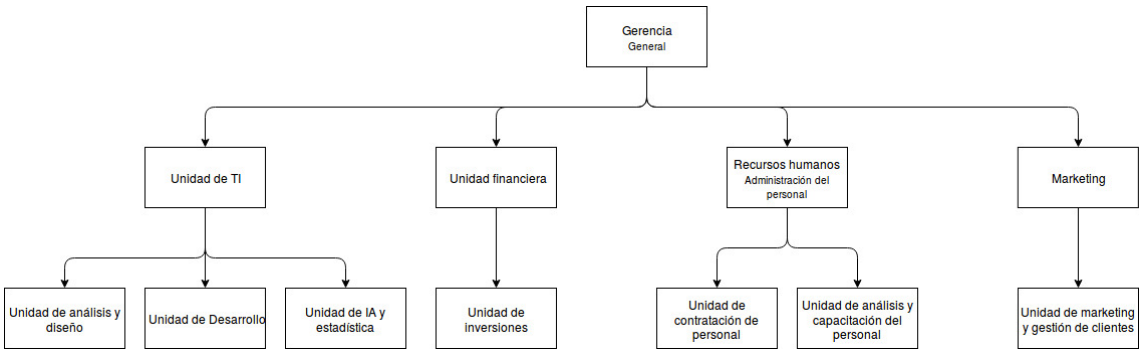

Figure 2.1: Organigrama de la empresa

2.2.2 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.

2.2.3 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.

2.2.4 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.

2.3 Arquitectura del sistema

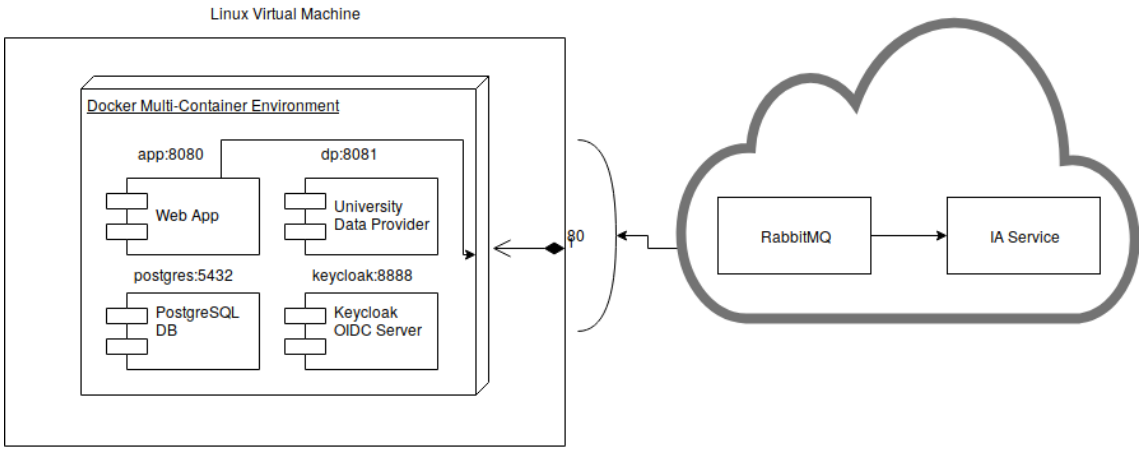

Figure 2.2: Arquitectura del Sistema

2.4 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.

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.5 Base de datos

A la hora de diseñar la base de datos hemos intentado construir un modelo que permita flexibilidad para poder adaptarnos a los distintos clientes y a sus requisitos. Para ello tenemos el siguiente modelo entidad relación. Se puede decir que el núcleo se encuentra en la relación entre curso, alumno y asignatura, creando una relación llamada matrícula donde se registrará al alumno con sus respectivas asignatura y curso. Con esto conseguimos tener una mayor flexibilidad a la hora de posibles casos individuales que puedan afectar a un sistema más fijo sin opción a modificaciones. Luego tenemos al profesor que podrá tanto impartir una o más asignaturas y a la vez coordinar un curso. También hemos definido una entidad para poder llevar los registros históricos del feedback semanal entregado por el alumno. Por último la entidad asignatura tendrá tantos exámenes, notas de comportamiento y entregables como sean requeridos. A continuación en el modelo relacional veremos qué atributos completan cada entidad. La relación matrícula creará una nueva entidad con las claves primarias de las entidades que crean dicha relación. Del curso se guardará su nombre, el coordinador y una breve descripción del mismo. El alumno tendrá datos personales para identificarlo y 'repetidor' que será un booleano para saber si no tiene las asignaturas estándar de un curso o puede que esté matriculado en más de uno. Luego la asignatura guardará los datos necesarios para identificarla, peso correspondiente en el curso y se completa tanto con los entregables, exámenes y notas de comportamiento que guardarán la fecha de cuando se registran, la competencia a la que pertenecen, y datos de identificación. Por último, para registrar los feedback de alumnado nos basta con guardar la fecha y los tres atributos recogidos en el feedback.

3. CHAPTER

Organización del Proyecto

3.1 Entorno de Desarrollo

3.1.1 Entorno Virtual de Desarrollo Docker

El entorno de desarrollo de la aplicación debe de 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 aportará Docker es el entorno uniforme entre todos los desarrolladores así como el entorno de producción y testeo. Además de facilitar las de-

pendencias de desarrollo como base de datos para testeo por ejemplo. Docker permite desplegar entornos completos sin tener que instalar software adicional.

El entorno funcionará 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 utilizará una extensión de Docker, docker-compose. La idea por detrás de esta herramienta es básicamente 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 puedan comunicarse entre sí. Se podría hacer lo mismo utilizando solamente Docker pero sería algo muy manual y con altas probabilidades de error.

Entrando en más detalles sobre qué contenedores se utilizarán, primero está el de la aplicación. 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 puede seguir actualizado con cada nueva iteración con 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 moría 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 contene-

dores son efímeros. Una vez el entorno es terminado todos cambios hechos al contenedor son borrados y si se intenta desplegar el entorno una vez más 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.

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:

```
version: "2.1"

services:
  app:
    image: maven:3-jdk-8-alpine
    ports:
      - 8080:8080
    volumes:
      - ./app:/app:delegated
      - ~/.m2:/root/.m2:delegated
    working_dir: "/app"
    command: ["sh", "redeploy.sh"]
  postgres:
    image: postgres:9.6
    environment:
      - POSTGRES_DB
      - POSTGRES_PASSWORD
```

Figure 3.1: Descripción del Entorno de Desarrollo

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

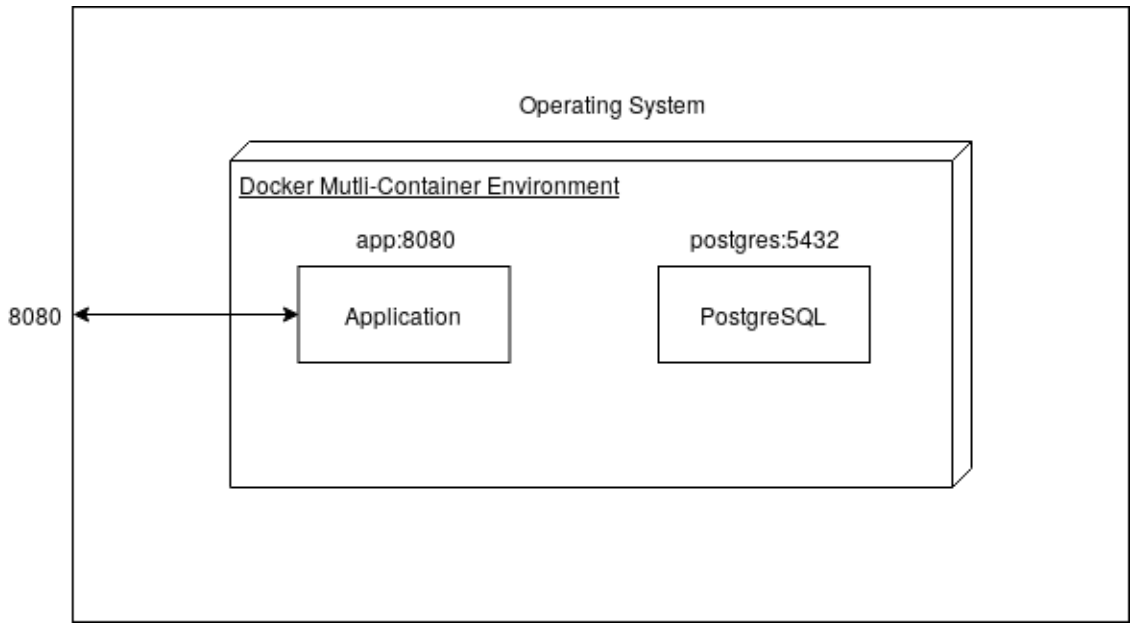

Figure 3.2: Entorno de Desarrollo en contenedores

3.1.2 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 desarrolladores. 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 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.

3.1.3 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.

3.1.4 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!

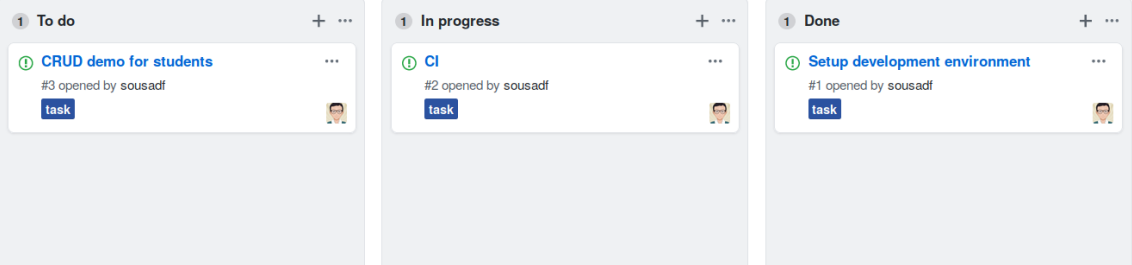

Figure 3.3: Kanban Board

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.

3.1.5 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.

3.1.6 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 .

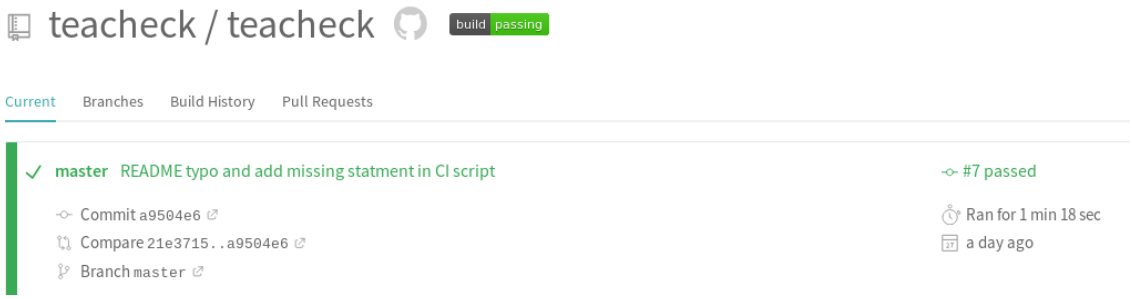

Figure 3.4: Pipeline status example

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 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).

3.1.7 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);