Actas de las Jenui, vol. 6. 2021.
Páginas: 235-242

Silence: un framework de apoyo a la docencia de desarrollo web

Agustín Borrego, Daniel Ayala, Fernando Sola, Inma Hernández, David Ruiz
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
41012 Sevilla
{borrego, dayala1, fsola, inmahernandez, druiz}@us.es

Resumen

La programación web es, actualmente, una de las vertientes más importantes de la informática en la industria nacional e internacional. Un aspecto muy relacionado, aunque en muchas ocasiones impartido de manera separada, es la enseñanza sobre bases de datos y el acceso a los datos contenidos en las mismas.

Con objeto de unir más estrechamente la enseñanza de estas dos ramas, en el presente trabajo presentamos Silence, un framework que provee un entorno unificado para el desarrollo de bases de datos relacionales que den soporte a una aplicación, y de aplicaciones web que consuman los servicios provistos por la base de datos a través de una API RESTful moderna.

Silence permite simplificar en gran medida el despliegue de endpoints RESTful sobre una base de datos relacional, posibilitando un aprendizaje del desarrollo web más ágil y una diferenciación clara entre back-end y front-end. Además, da soporte al aprendizaje basado en proyectos, siendo éstos autocontenidos y facilitando su despliegue, desarrollo y evaluación.

El framework se encuentra actualmente en uso por más de 500 alumnos en tres titulaciones de grado, y está disponible libremente como código abierto.

Abstract

Web development is currently one of the most prominent fields in the computing industry, both local and international. Database management and usage stands out as a highly related field, though it is often taught independently.

In this work we introduce Silence, a framework whose goal is to allow for a joint teaching of databases and web development. Silence provides a unified environment for developing relational databases to store an application's data, and web applications that use such databases through a modern RESTful API.

Silence greatly simplifies the process of deploying RESTful endpoints to interact with a database, resulting in a more agile web development learning process and in a very clear distinction between back-end and front-end. Furthermore, Silence is oriented towards project-based learning, offering self-contained projects that are easy to develop, deploy and evaluate.

Our framework is currently actively used by more than 500 students in three different degrees, and it is freely available as open source software.

Palabras clave

Back-end, front-end, APIs REST, desarrollo web, framework.

1. Introducción

El desarrollo web sigue siendo una de las áreas de conocimiento más demandadas en el entorno empresarial, siendo los perfiles más demandados los de desarrollador back-end, front-end o full-stack. Incluso en perfiles que no son estrictamente de desarrollador web, es frecuente encontrar el conocimiento de estas tecnologías como uno de los requisitos deseables. En España, estas competencias suelen cubrirse mediante dos asignaturas diferentes.

Sin embargo, el rápido ritmo de evolución de las tecnologías web hace difícil mantener actualizado el currículum académico de las asignaturas relacionadas sin caer en algunos errores comunes: por ejemplo, dedicar demasiado tiempo a que los alumnos aprendan muchas tecnologías diferentes, mientras que no llegan a asimilar los conceptos básicos y más perdurables de la materia (los que, a la larga, más útiles pueden resultarles), o diseñar asignaturas que intentan enseñar las más recientes tecnologías y quedan obsoletas en no demasiado tiempo [2].

El diseño de un stack tecnológico adecuado para este tipo de asignaturas, por tanto, pasa por lograr un equilibrio entre tecnologías actuales y que vayan a resultar útiles a los alumnos de forma inmediata en su vida

profesional (Python, JavaScript, AJAX, CSS, HTML, SQL, APIs REST) y darles unos fundamentos teórico-prácticos sólidos de una duración más a largo plazo (trazabilidad entre los requisitos, el modelado y la base de datos). No está claro dónde debería estar el punto óptimo de equilibrio, y es por ello que no existe un consenso claro (ni en cuanto a tecnologías ni en cuanto a bibliografía) sobre cómo diseñar estas asignaturas. Lo único que parece claro es que, por su naturaleza, la metodología que se use en ellas debería ser aprendizaje basado en proyectos (ABP) [1]. Algunos autores consideran que la clave para tomar esta decisión está en el punto de madurez de los alumnos, siendo conveniente simplificar el stack tecnológico a los alumnos de primeros cursos y añadiendo complejidad a medida que van pasando a cursos superiores [5].

Nuestra propuesta se enmarca en la experiencia en dos asignaturas de 2º curso en la Universidad de Sevilla, con alumnos que únicamente han adquirido conceptos básicos de programación en una asignatura previa, por lo que un enfoque tecnológico más simple está justificado. Con este fin, hemos desarrollado Silence, una herramienta integral para el desarrollo de aplicaciones web, que permite a los alumnos implementar rápidamente el back-end y/o el front-end de estas aplicaciones sin necesidad de conocimientos avanzados de programación y sin depender de frameworks destinados a uso más profesional que puedan quedar obsoletos, además de resultar difíciles de usar para alumnos de segundo curso. Las principales ventajas de Silence son, entre otras: la reducción en el tiempo de desarrollo frente a frameworks más complejos, la reducción en la curva de aprendizaje de las tecnologías que permite dedicar más tiempo del currículum a fundamentos teóricos básicos, la posibilidad de implementar de forma sencilla pruebas y la estabilidad del stack tecnológico, frente a frameworks más volátiles. Silence se ha implantado en tres titulaciones de grado con un total de 524 alumnos, y las impresiones de alumnos y profesores son muy positivas.

El resto del artículo se organiza de la siguiente forma: en la Sección 2 describimos las asignaturas en las cuales se contextualiza el uso de Silence, en la Sección 3 analizamos el trabajo relacionado, en la Sección 4 describimos la estructura del framework y los proyectos Silence, en la Sección 5 presentamos Noise, una herramienta relacionada, para después en la Sección 6 presentar los resultados obtenidos en su implantación; finalmente, la Sección 7 concluye el artículo.

2. Contexto docente

Este framework se ha desarrollado en el contexto de dos asignaturas obligatorias cuatrimestrales en la Universidad de Sevilla: Introducción a la Ingeniería del Software y los Sistemas de Información 1 (en adelante, IISSI-1) y 2 (en adelante, IISSI-2), ambas de segundo curso.

IISSI-1 e IISSI-2 abordan dos aspectos complementarios del desarrollo de aplicaciones web, que se corresponden con lo que actualmente se denominan back-end y front-end. La metodología de las dos asignaturas se engloba dentro del conocido como aprendizaje basado en proyectos que ha sido objeto de análisis en muchas asignaturas de nuestra área [4, 6].

La asignatura IISSI-1 cubre todo el proceso de desarrollo de bases de datos relacionales. Esta cuenta con cuatro bloques de contenidos: el primero de ellos sirve para introducir los conceptos básicos relacionados con la Ingeniería del Software y el análisis de requisitos; el segundo se centra en el diseño de bases de datos relacionales normalizadas a partir de una especificación de requisitos; en el tercero se aborda la implementación de una base de datos a partir de un modelo relacional usando SQL; por último, en el cuarto bloque se estudia cómo definir endpoints para interactuar con la base de datos a través de una API REST, y cómo realizar pruebas usando esta misma API.

La asignatura IISSI-2 aborda los aspectos relacionados con el proceso de desarrollo de una interfaz web que pueda consumir datos procedentes de una o varias APIs REST. La asignatura cuenta con tres bloques de contenidos: el primero de ellos introduce las aplicaciones web y cómo diseñar interfaces web teniendo en cuenta aspectos de usabilidad, accesibilidad y experiencia de usuario; el segundo se centra en el desarrollo de interfaces estáticas usando HTML y CSS; finalmente, el tercer bloque se centra en la implementación de comportamiento dinámico usando JavaScript.

La evaluación en ambas asignaturas se lleva a cabo mediante diferentes entregas en la que el alumno ha de desarrollar incrementalmente su proyecto. Una vez el alumno ha llevado a cabo todas las entregas requeridas puede optar a la defensa de su proyecto, en la que se plantean cuestiones sobre los elementos desarrollados y se pide realizar modificaciones sobre los mismos.

3. Trabajo relacionado

Un análisis de las asignaturas existentes sobre programación web reveló que el conjunto de tecnologías necesarias es tan amplio que no es factible abordarlo en una única asignatura; lo habitual es dividir los contenidos en, al menos, dos asignaturas: una dedicada al back-end y otra al front-end [2, 3, 5].

Esto hace que, por ejemplo, al estudiar la asignatura de front-end, la parte de back-end ha de ser simulada o se ha de desplegar una versión de prueba muy sencilla, que permita a los alumnos centrarse en profundizar en los conceptos de front-end (y de forma similar ocurre

a la inversa). Actualmente, las herramientas de simulación de back-end más usadas son:

- JSON Server: Permite desplegar endpoints CRUD sobre recursos definidos en un archivo JSON. La implementación de reglas de negocio, por sencillas que sean, es poco intuitiva, y la gestión de sesiones requiere de un plugin separado.
- MockServer: Asigna respuestas determinadas a peticiones concretas, o puede actuar como proxy para peticiones más complejas. Tiene los mismos problemas que la alternativa anterior.
- Mocky: También permite responder de forma determinada a una petición. Su uso más allá de unos determinados márgenes es de pago.

En el caso de los front-end de prueba, las alternativas son mucho menos numerosas, debido a la mayor dificultad inherente a crear de manera automática una interfaz de usuario de prueba con el nivel mínimo de complejidad necesario.

La alternativa es usar frameworks integrales, que combinen el desarrollo back-end con front-end, cubriendo así todos los aspectos relevantes del desarrollo web. Sin embargo, los frameworks web más usados actualmente (Django, Spring, Laravel, Rails...) imponen un patrón de diseño MVC y requieren un dominio de la tecnología que entra en conflicto con los conceptos básicos que pretendemos transmitir a nuestros alumnos de segundo curso. Incluso aquellos frameworks web populares más sencillos, como Flask para Python, no son alternativas factibles al requerir de aprendizaje especializado, cuyo tiempo podría ser dedicado a la enseñanza de conceptos básicos. Además, las operaciones contra la base de datos quedan demasiado abstraídas para la docencia. Es por esto que otros autores han desaconsejado el uso de frameworks complejos en asignaturas de programación web, principalmente en los primeros cursos [5].

4. Silence

Silence es un framework destinado a facilitar el desarrollo ágil de una API RESTful capaz de ofrecer operaciones sobre recursos almacenados en una base de datos relacional, y de una aplicación web basada en el uso de dichos endpoints con una clara separación entre back-end y front-end. Dado que este conjunto de esquema de base de datos, endpoints REST y archivos web definen una aplicación, Silence hace uso de un enfoque basado en proyectos que unifica y estandariza estos aspectos, y permite un despliegue sencillo mediante una interfaz de línea de comandos. A continuación, se detallan la arquitectura del framework y la estructura de un proyecto Silence.

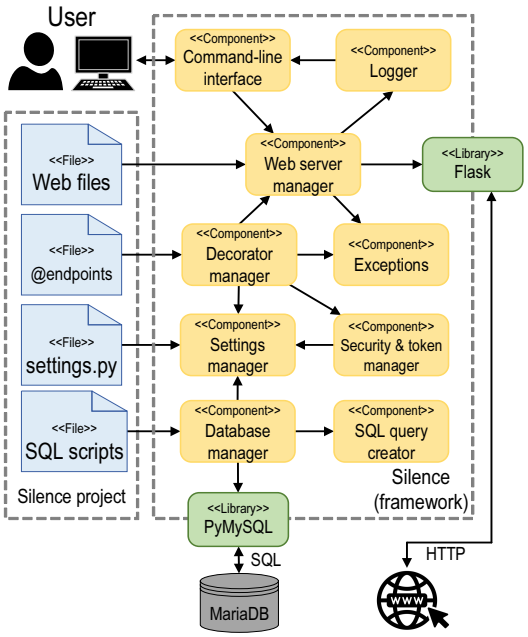

Figura 1: Arquitectura del framework y los proyectos Silence

4.1. Arquitectura de Silence

Silence está desarrollado en Python mediante una arquitectura basada en componentes, y su código fuente está disponible en GitHub[1]. También se encuentra disponible en PyPI[2], el repositorio de paquetes de Python, para su fácil instalación y actualización. Cada componente es responsable de gestionar ciertos aspectos del framework, y expone una interfaz al resto de componentes que requieran sus servicios.

El esquema general de la arquitectura de Silence, y cómo ésta se relaciona con cada uno de los aspectos de un proyecto Silence, se encuentra en la Figura 1. Cada componente se explica brevemente a continuación:

Interfaz de línea de comandos (CLI): Este componente es responsable de proveer una interfaz de consola para la creación, gestión y despliegue de proyectos Silence. Los comandos disponibles son:

- `silence new`: Crea un nuevo proyecto a partir de una plantilla o la URL de un repositorio. Si no se especifica una plantilla, se usa una por defecto.
- `silence list-templates`: Muestra las plantillas de proyectos disponibles.
- `silence createdb`: Efectúa una conexión a la base de datos y ejecuta los scripts SQL incluídos en el proyecto para crear la estructura de tablas e insertar datos iniciales.

[1] `https://github.com/IISSI-US/Silence`
[2] `https://pypi.org/project/Silence/`

- `silence run`: Ejecuta el servidor que da soporte a los endpoints de la API definida en el proyecto y a los archivos web.

Logger: El componente de logging es responsable de estandarizar los mensajes emitidos por las librerías usadas internamente, filtrar aquellos que no son relevantes, y emitir salidas acordes al nivel de log deseado.

Gestor del servidor web: Se encarga de realizar en él la configuración oportuna según los ajustes del proyecto, de cargar los endpoints definidos en el proyecto, y de proveer una serie de endpoints por defecto (salvo que éstos sean desactivados en la configuración del proyecto). Estas funcionalidades por defecto incluyen el registro de nuevos usuarios, el inicio de sesión de usuarios existentes y el listado de todos los endpoints disponibles en el servidor.

Gestor de decoradores: Este componente provee este tipo de objetos, que son utilizados a lo largo del framework para diversas funciones. Uno de los decoradores más relevantes es `@endpoint`, empleado por el usuario para definir endpoints en sus proyectos.

Gestor de excepciones: Es responsable de proveer un conjunto unificado de excepciones, así como de capturar las mismas cuando se producen en distintos puntos del framework para mostrar mensajes de error adecuados.

Gestor de ajustes: Provee una interfaz unificada a la configuración del framework a todos los demás componentes. Además, es el responsable de cargar la configuración individual de cada proyecto, así como de proveer valores por defecto para aquellos parámetros de configuración que no estén definidos en un proyecto.

Gestor de tokens y seguridad: Tiene como función primaria emitir y comprobar *tokens* de sesión, que representan la sesión iniciada de un usuario y que tienen un determinado período de caducidad. Además, es el encargado de proveer una interfaz para almacenar y validar contraseñas de manera criptográficamente segura.

Gestor de la base de datos: Provee una interfaz programática abstracta respecto a la base de datos empleada, facilitando la compatibilidad de Silence con otros posibles sistemas gestores de bases de datos en el futuro. También, lee y ejecuta los scripts SQL del proyecto en la base de datos seleccionada cuando lo solicita el usuario mediante la interfaz de línea de comandos.

Creador de queries SQL: Por último, este componente permite generar este tipo de consultas programáticamente, facilitando una gestión abstracta de la base de datos por parte del componente anterior.

4.2. Estructura de un proyecto Silence

Cada proyecto Silence está compuesto de una serie de elementos principales, que interactúan con los distintos componentes del framework según lo mostrado en la Fig. 1. En las siguientes subsecciones se pasa a enumerar y explicar cada uno de estos elementos. Como ejemplo, supongamos un proyecto dedicado a la gestión de los departamentos de una universidad. Este proyecto de ejemplo está disponible en GitHub[3].

Endpoints: Una de las motivaciones en el desarrollo de Silence es permitir la definición de endpoints de una API RESTful de manera sencilla. La forma de definir endpoints en un proyecto Silence es creando un archivo Python por cada entidad del dominio que se desea que sea accesible mediante la API, dentro de la carpeta `endpoints/` del proyecto. En estos archivos, se pueden definir tantos endpoints como sean necesarios.

El decorador `@endpoint` es usado para definir un punto de acceso a la API. En él se definen la ruta del recurso, el método HTTP a emplear y la consulta SQL asociada. El resultado de la consulta será convertido automáticamente a JSON y devuelto al cliente. Es necesario asociar a cada definición de endpoint una función Python, que puede usarse para validar parámetros recibidos (ver Figura 3). En caso de no realizar ninguna validación, como es el caso de una consulta GET, la función puede no realizar ninguna operación.

También es posible definir rutas con parámetros, que pueden ser capturados y usados en la sentencia SQL asociada al endpoint. A modo de ejemplo, la Figura 2 muestra cómo definir un endpoint que permite acceder a un departamento dado su identificador.

```
@endpoint(
    route="/departments/$depId",
    method="GET",
    sql="SELECT * FROM
        Departments WHERE
        departmentId = $depId"
)
def get_by_id():
    pass
```

Figura 2: Definición de un endpoint GET con parámetros de ruta

Las partes de una ruta que comienzan por $ se capturan como parámetros, y pueden ser insertados en la consulta SQL. Estos parámetros se reemplazan de forma segura, evitando ataques por inyecciones SQL.

Los endpoints pueden aceptar también datos enviados en el cuerpo de una petición, por ejemplo de un formulario, tanto en formato URL como JSON. Para especificar que se esperan parámetros en el cuerpo de la petición, éstos se incluyen como parámetros de la función asociada y pueden ser accedidos en ella para su

[3] https://github.com/IISSI-US/Silence-template-Employees

validación. Por ejemplo, en la Figura 3 se muestra cómo implementar la creación de un nuevo departamento dados su nombre y su ciudad mediante una petición POST, comprobando que el nombre del departamento tenga al menos 3 caracteres:

```
@endpoint(
   route="/departments",
   method="POST",
   sql="INSERT INTO Departments (
       name, city) VALUES ($name,
       $city)"
)
def insert(name, city):
   if len(name) < 3:
       raise HTTPError(400, "The
           name is too short")
```

Figura 3: Definición de un endpoint POST

En el caso de endpoints que no sean de tipo consulta, el valor devuelto es el identificador del último recurso creado o modificado.

La definición de otras operaciones es análoga, pudiendo crear endpoints DELETE para el borrado de recurso o PUT para su actualización. En este último caso, se pueden combinar parámetros de ruta y parámetros recibidos en el cuerpo de la petición HTTP.

Archivos SQL: Cada proyecto Silence contiene una carpeta `sql/` que incluye todos los archivos SQL necesarios para crear la base de datos del proyecto: tablas, datos iniciales, disparadores, procedimientos, etc. Esto permite que los proyectos sean autocontenidos, dado que tienen toda la información necesaria para establecer la base de datos a un estado listo para su uso mediante el comando `silence createdb`.

El orden de ejecución de los scripts SQL cuando se invoca la orden anterior se define en la configuración del proyecto.

Configuración del proyecto: Los proyectos contienen un archivo `settings.py` que gobierna un gran número de aspectos relacionados con la configuración del mismo, tales como los datos de acceso al SBGD, el orden de ejecución de los scripts SQL para la creación de la base de datos, el puerto del servidor web, el control del nivel de log, etcétera. El listado completo de los parámetros de configuración disponibles se encuentra en la documentación online de Silence.

Archivos web: La carpeta `docs/` del proyecto es la raíz de los archivos HTML, CSS y JavaScript que implementan la interfaz web de la aplicación, además de ficheros relacionados como imágenes, fuentes, etc.

Los proyectos plantilla proporcionados siguen una filosofía de estructuración del código web inspirada en un esquema basado en componentes, que facilita el desarrollo de código ordenado y reutilizable. Los aspectos clave son los siguientes:

- Cada vista se implementa mediante un archivo `.html`.
- Cada archivo `.html` tiene asociado un archivo `.js` homónimo en la carpeta `docs/js/`, responsable de gestionar los eventos de la vista y de proporcionar el punto de entrada.

Asimismo, el código JavaScript de la aplicación, localizado en la carpeta `docs/js/`, se organiza de la siguiente manera:

- La carpeta `api/` contiene módulos JavaScript encargados de comunicarse con la API del proyecto mediante una interfaz basada en promesas[4].
- La carpeta `renderers/` contiene módulos JavaScript cuya función es transformar las diferentes entidades del dominio del problema en representaciones HTML adecuadas.
- La carpeta `utils/` contiene módulos JavaScript con utilidades de propósito general.
- Por último, la carpeta `validators/` contiene módulos responsables de la validación de los formularios de la aplicación.

5. Noise

La homogeneidad en los proyectos y su despliegue nos ha permitido desarrollar, además, una herramienta llamada Noise, que permite a los estudiantes desplegar sus aplicaciones web en un servidor habilitado para tal fin. Noise es, a su vez, un proyecto Silence, cuya API permite a los alumnos desplegar otros proyectos Silence en el mismo servidor, y que posee una interfaz de administración web para el profesorado destinada a gestionar y monitorizar los proyectos del alumnado.

En las secciones siguientes procedemos a explicar el proceso de despliegue y autogestión de proyectos por parte de los alumnos, así como las herramientas puestas a disposición de los profesores.

5.1. Despliegue de proyectos

La API de Noise provee un endpoint `/deploy` que permite desplegar un proyecto Silence en el servidor indicando la URL del repositorio que lo contiene, como se ilustra en la Figura 4.

Si el servidor se encuentra en condiciones de atender la petición de despliegue (comprobando, por ejemplo, que el alumno tiene permitido usar el servicio), responderá con un código HTTP 202 (aceptado), incluyendo

[4] `https://developer.mozilla.org/en-US/docs/web/JavaScript/Reference/Global_Objects/Promise`

```
POST [...]/api/deploy
Content-Type: application/json
{
    "url": "https://github.com/
       alumno/repo_proyecto.git"
}
```

Figura 4: Petición de despliegue de proyecto

un identificador que el alumno puede usar para gestionar su proyecto en el servidor. Un ejemplo de esta respuesta se muestra en la Figura 5.

```
HTTP/1.1 202 ACCEPTED
{
  "code": 202,
  "message": "Accepted",
  "projectId": "
     AdZpPRx4KN8FSOYcEv6VuCY8O"
}
```

Figura 5: Respuesta a la solicitud de despliegue

Se puede consultar el estado de un proyecto desplegado a partir de su identificador, realizando una petición GET al endpoint `/projects/<id>`. La respuesta del servidor dependerá del estado en el que se encuentre el proyecto en cuestión. Como ejemplo, se muestra en la Figura 6 una posible respuesta por parte de Noise a una petición de estado del proyecto desplegado anteriormente, si éste se encuentra en ejecución sin incidencias. En ella, se incluye una URL al propio servidor en la que se puede acceder a la interfaz web del proyecto desplegado, al que le ha sido asignado un puerto determinado del servidor.

El endpoint `/projects/<id>` también puede ser usado para detener un proyecto desplegado, mediante una petición HTTP de tipo DELETE. Cuando un proyecto se detiene, su código fuente se comprime y almacena, y se retiene la información sobre su despliegue para futuras consultas.

5.2. Interfaz web

Al tratarse de un proyecto Silence, Noise también cuenta con una interfaz web. El propósito de ésta es facilitar la consulta y gestión por parte de los profesores de los proyectos desplegados por los alumnos, y mostrar para cada uno de ellos una serie de estadísticas e información de interés.

Esta interfaz está reservada a los profesores, que deberán contar con un usuario ya registrado en la base de datos (el registro de nuevos usuarios está desactivado

```
HTTP/1.1 200 OK
{
  "code": 200,
  "message": "OK",
  "project": {
    "date": "Sun, 07 Feb 2021
       16:58:44 GMT",
    "projectId": "
       AdZpPRx4KN8FSOYcEv6VuCY8O"
       ,
    "repo": "https://github.com/
       alumno/repo_proyecto.git",
    "status": "RUNNING",
    "url": "http://[...]:5532"
  }
}
```

Figura 6: Respuesta a la consulta de estado del proyecto desplegado

en Noise). Al entrar, podrán consultar un listado de los proyectos activos en el servidor en ese momento junto con la información básica de cada uno de ellos: puerto en el que está desplegado, identificador, URL del repositorio del alumno, nombre del alumno, y número de *commits* en GitHub del proyecto, así como el número de veces que el proyecto ha sido desplegado en el servidor usando Noise anteriormente. El profesor puede ordenar los proyectos y realizar búsquedas por cualquiera de estos parámetros. Además, se dispone de un listado similar de proyectos desplegados anteriormente en el servidor, pero detenidos por cualquier razón (fallo en el código del proyecto, detención manual por parte del alumno...), para poder conocer el histórico de despliegues realizados en el servidor.

Al acceder a los detalles de cualquier proyecto se muestra información adicional sobre el mismo, como asistencia al profesor para la evaluación del alumno. En esta información se incluyen datos como el número total de *commits* en GitHub, el número de líneas de código añadidas y eliminadas hasta el momento, la distribución de los *commits* a lo largo del tiempo (Fig. 7) y el detalle de todos ellos junto con su fecha y comentario. Otra información relevante mostrada es el número de despliegues anteriores usando Noise a lo largo del tiempo y las fechas de los mismos, junto con la causa que motivó la detención del proyecto en el servidor para ese despliegue (Fig. 8).

6. Evaluación y resultados

Para evaluar la satisfacción del alumnado con el uso de Silence, se les pidió cumplimentar de manera vo-

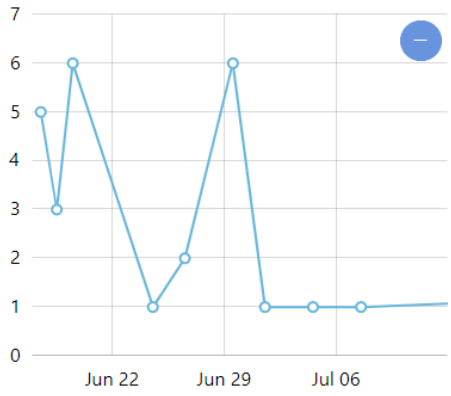

Figura 7: Distribución temporal de *commits* Git de un proyecto Silence desplegado usando Noise

Show 10 entries

Search:

Creation Date	Stop Date	Stop Cause
Tue, 02 Feb 2021 17:18:51 GMT	Tue, 02 Feb 2021 17:19:15 GMT	ADMIN_STOP
Tue, 02 Feb 2021 17:20:21 GMT	Tue, 02 Feb 2021 17:20:22 GMT	USER_STOP
Tue, 02 Feb 2021 17:20:49 GMT	Tue, 02 Feb 2021 17:20:51 GMT	USER_STOP
Tue, 02 Feb 2021 17:27:06 GMT	Tue, 02 Feb 2021 17:28:46 GMT	REPLACED

Figura 8: Listado de despliegues anteriores en Noise para un determinado proyecto

luntaria y anónima una encuesta evaluando diferentes aspectos del framework y su experiencia personal con él. La encuesta fue remitida a 6 grupos de laboratorio en total, repartidos por las diferentes titulaciones y agrupando a un total de 172 alumnos, de los cuales 42 respondieron a la encuesta.

En el cuestionario, se les pidió que valoraran de menor (1) a mayor (5) grado de satisfacción los siguientes aspectos sobre Silence:

- Facilidad de uso y/o instalación.
- Facilidad de configurar la conexión a una base de datos.
- Facilidad de despliegue de endpoints.
- Claridad y ayuda de los mensajes de error.
- Ayuda para comprender y poner en práctica los conceptos teóricos sobre APIs RESTful.

Además, se les pidió que comentaran mediante respuesta libre aquellos aspectos en los que habían encontrado dificultades, y los cambios que sugerirían hacer.

Una encuesta similar fue remitida al profesorado de la asignatura, para que remitieran de forma anónima su opinión sobre el uso de Silence como docentes. Las preguntas sobre satisfacción son casi idénticas, con la excepción de que en la última de ellas se pide responder si, en su opinión, creen que es una buena herramienta para explicar y poner en práctica el contenido teórico.

Los resultados de esta encuesta, tanto para alumnos como para profesores, se muestran en la Fig. 9, mientras que las respuestas más comunes a las preguntas de texto libre se discuten a continuación.

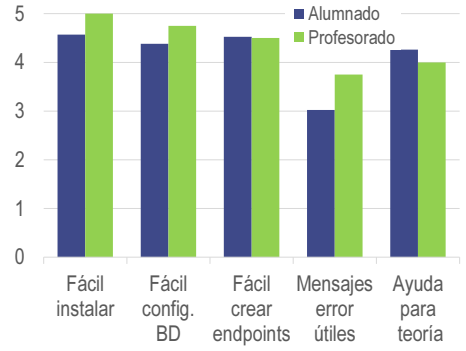

Figura 9: Resultados medios de la encuesta de satisfacción

A tenor de los resultados obtenidos en la encuesta de satisfacción, mostrados en la Fig. 9, podemos concluir que la experiencia de uso de Silence ha sido positiva tanto para el alumnado como para el profesorado. Los datos recopilados muestran, en general, que Silence es fácil de instalar, configurar y usar para desplegar endpoints con los que dar soporte a una API RESTful.

Los usuarios han valorado, además, que se trata de una herramienta útil tanto para enseñar como para poner en práctica los conceptos teóricos impartidos en la asignatura. Creemos que esto es una consecuencia de lo anterior, dado que su facilidad y rapidez de uso permiten a los profesores hacer demostraciones en directo y a los alumnos seguir esas demostraciones en sus propios ordenadores. Así, pueden probar por ellos mismos de manera dinámica la definición de endpoints, y experimentar con posibles cambios.

El cuestionario incluía, también, secciones de texto libre en la que los alumnos y profesores podían hacer comentarios sobre los aspectos a mejorar en el framework y proponer cambios. En general, las respuestas recibidas versaban sobre dos aspectos principales: la gestión de errores y el uso de la consola de comandos como forma de interacción con Silence. Como se aprecia en la Fig. 9, la utilidad y claridad de los mensajes de error ha sido el aspecto peor valorado en general, con un 3.02 sobre 5 de nota media. En sus comentarios, algunos alumnos indicaron que determinados mensajes

de error pueden ser poco intuitivos, no siendo obvio a qué son debidos y cómo solucionarlos. Esta percepción es compartida, aunque en menor medida, en las respuestas emitidas por el profesorado. Por otro lado, el uso de comandos en terminal, actualmente la única forma de interactuar con Silence para iniciar el servidor o realizar la carga inicial de la base de datos, ha resultado confuso para algunos alumnos. Aunque la cantidad y complejidad de los comandos de Silence es reducida, en algunos casos ha supuesto la primera interacción del alumnado con una herramienta de línea de comandos, añadiendo una carga cognitiva adicional.

En futuras versiones de Silence nos centraremos en corregir estos defectos, mejorando la calidad y claridad de los mensajes de error y desarrollando una interfaz gráfica con la que poder interactuar con el framework. Otras mejoras adicionales a desarrollar incluyen permitir definir endpoints mediante un lenguaje más abstracto, no necesariamente usando la sintaxis del lenguaje Python; y la posibilidad de escribir tests automatizables para cada proyecto. Finalmente, una vez terminado el segundo cuatrimestre, evaluaremos de manera similar la experiencia del alumnado con el apartado relativo al desarrollo web y a la estructura de archivos web propuesta por Silence, e identificaremos posibles aspectos de mejora en este sentido.

Por último, puesto que Silence está disponible como un paquete Python en PyPI, el índice de paquetes del lenguaje, hemos recopilado estadísticas sobre el número de descargas que ha tenido a lo largo del tiempo gracias a la herramienta PePy[5]. Desde su publicación inicial y a fecha de la redacción del presente artículo, Silence ha sido descargado un total de 37.153 veces, con una media de alrededor de 100 descargas al día.

7. Conclusiones

En este artículo hemos presentado Silence, un framework orientada a proyectos y destinada a facilitar el desarrollo web. Silence cubre tanto el back-end, mediante un despliegue sencillo de endpoints que implementan una API RESTful sobre una base de datos relacional; como el front-end, incorporando los archivos de la aplicación web en el propio proyecto y recomendando una estructura de archivos HTML, CSS y JS que simplifica la gestión del código. La estandarización de la estructura de los proyectos, y la facilidad de su gestión y despliegue a través de la línea de comandos, nos ha permitido desarrollar otra herramienta adicional, Noise, cuyo objetivo es facilitar a los alumnos el despliegue de sus proyectos en un servidor de la Universidad.

En este primer cuatrimestre, más de 500 alumnos en tres titulaciones diferentes han usado Silence para poner en práctica los conceptos sobre servicios RESTful y APIs aprendidos. Para conocer el grado de satisfacción con el framework, invitamos tanto a alumnos como a profesores a cumplimentar anónimamente una encuesta valorando diferentes aspectos de Silence. Los resultados han sido en general muy positivos, remarcando la facilidad de instalación y uso del framework, así como su utilidad para explicar y mostrar los elementos fundamentales de la asignatura.

[5] `https://pepy.tech/project/Silence`

Referencias

[1] Óscar Cánovas Reverte and Félix Jesús García Clemente. Prevención y seguimiento de factores limitantes del trabajo en equipo en experiencias ABP. In *Actas de las XXII Jornadas sobre la Enseñanza Universitaria de la Informática, Jenui 2016*, pages 11–18, 2016.

[2] Randy Connolly. Facing backwards while stumbling forwards: The future of teaching web development. In *SIGCSE'19*, pages 518–523, 2019.

[3] Ralph F. Grove. Trends in teaching web-based development a survey of pedagogy in web development courses. In *WEBIST*, pages 361–365, 2007.

[4] Pengyue Guo, Nadira Saab, Lysanne S. Post, and Wilfried Admiraal. A review of project-based learning in higher education: Student outcomes and measures. *International Journal of Educational Research*, 102:101586, 2020.

[5] Francesco Maiorana. Extending the database curriculum: From design principles to web and mobile programming. In *Computer Supported Education*, pages 258–271, 2015.

[6] Carlos J. Villagrá-Arnedo, Francisco J. Gallego-Durán, Faraón Llorens-Largo, and Rafael Molina-Carmona. Movimientos pendulares al situar al estudiante en el centro del proceso de aprendizaje. In *Actas de las XXII Jornadas sobre la Enseñanza Universitaria de la Informática, Jenui 2016*, pages 285–291, 2016.