Catalyde: Un entorno de iniciación a la programación con fines educativos

Catalin Costin Stanciu

Universitat Politècnica de València

25 de febrero de 2013

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escuela Técnica Superior de Ingeniería Informática

Índice

Presentación del problema

- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
  - Aprender a programar.
  - Aprender a usar el entorno.

Presentación del problema

- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
  - Aprender a programar.
  - Aprender a usar el entorno.

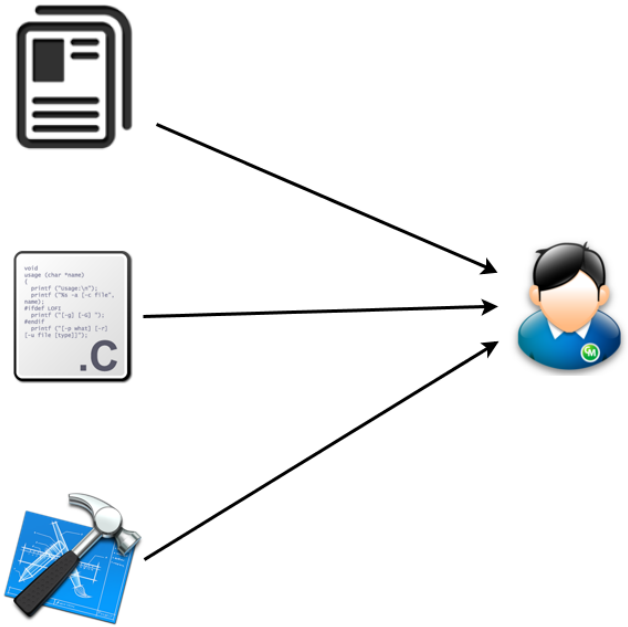
Presentación del problema

- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
  - Aprender a programar.
  - Aprender a usar el entorno.

Presentación del problema

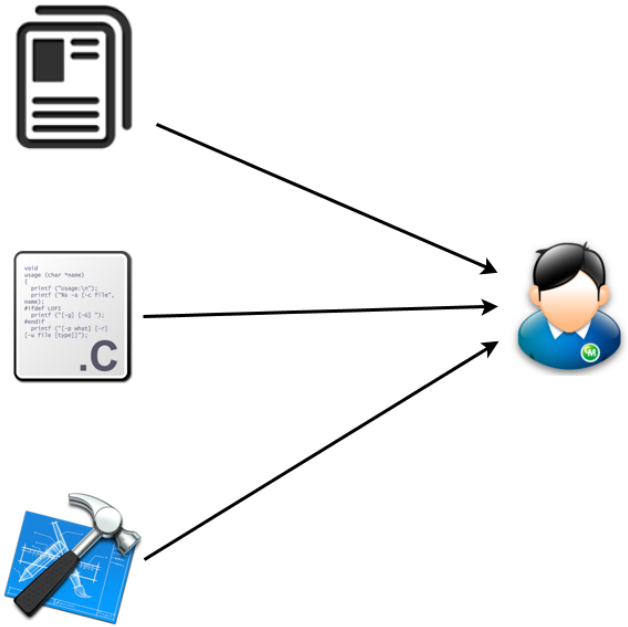
- Programar supone ser capaz de convertir un modelo mental de "*lo que tiene que hacer o resolver un programa*" en el diseño y la posterior implementación que lo realice.
- Los alumnos deben centrarse en la parte algorítmica y no distraerse con detalles.
- Entender lo que se pide en un ejercicio de programación es esencial y muchas veces los alumnos no entienden realmente lo que el profesor pide con un ejercicio.
- Doble esfuerzo al iniciarse en el mundo de la programación:
  - Aprender a programar.
  - Aprender a usar el entorno.

Estado actual



- El **profesor** proporciona al alumno una serie de contenidos:
  - Enunciado
  - Código para rellenar (opcional)
- El **alumno**:
  - Utiliza un entorno de desarrollo (más o menos complejo) para escribir y depurar el programa.
  - No se puede comprobar el funcionamiento hasta que se haya completado todo el programa.

Estado actual



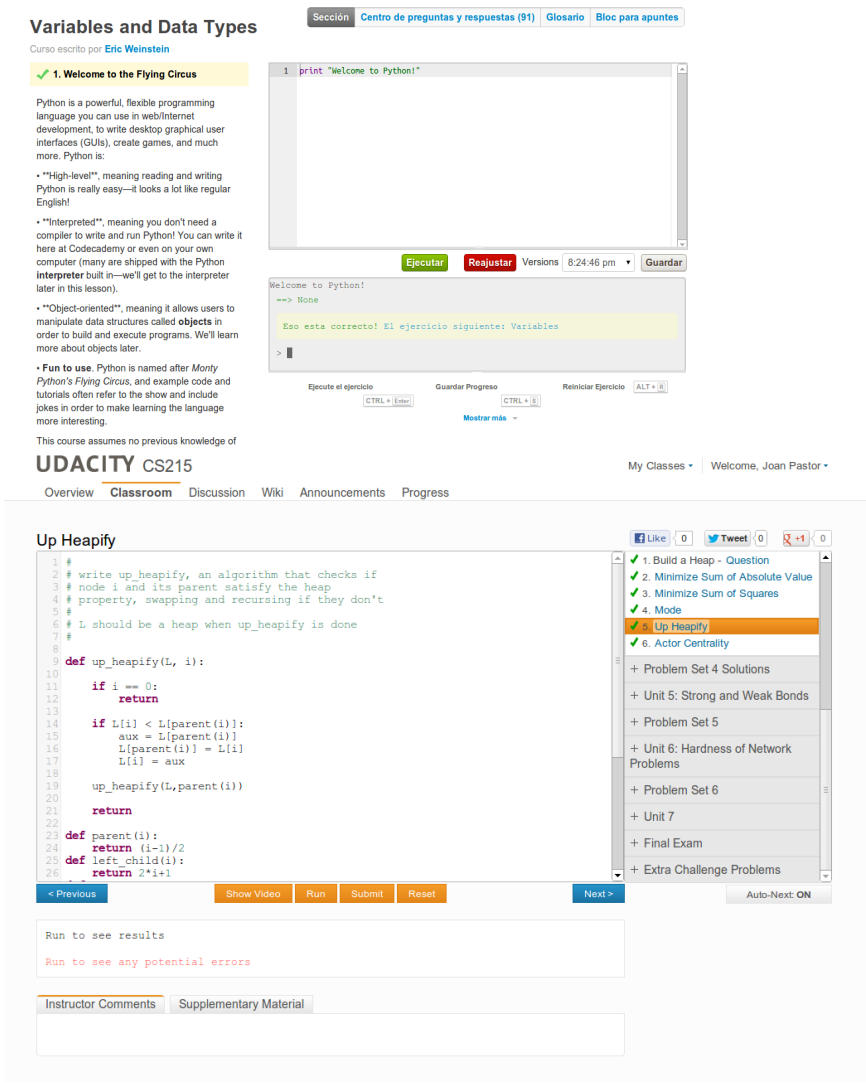
- El **profesor** proporciona al alumno una serie de contenidos:
  - Enunciado
  - Código para rellenar (opcional)
- El **alumno**:
  - Utiliza un entorno de desarrollo (más o menos complejo) para escribir y depurar el programa.
  - No se puede comprobar el funcionamiento hasta que se haya completado todo el programa.

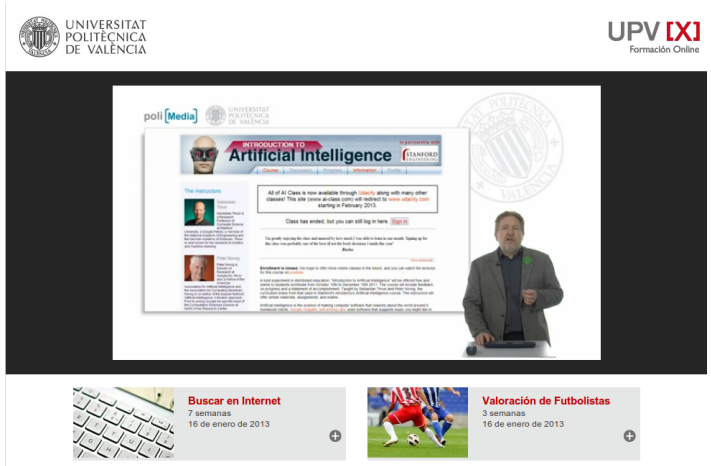
Cursos Abiertos Masivos En Línea

¿Qué son?

- Gran auge de los cursos a distancia debido a las plataformas *online*.
- Aparición de los Cursos Abiertos En línea Masivos (en inglés Massive Open Online Course(MOOC)).
- Existen varias plataformas (y cada vez más). Muchas relacionadas con la computación y, en concreto, con la programación.

Cursos Abiertos Masivos En Línea





Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.

Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.

Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.

Herramienta propuesta

- Catalyde: Catalyzer Development Environment.
- La aplicación Catalyde introduce a los usuarios (alumnos) en la disciplina de la programación.
- La herramienta está destinada a estudiantes de primeros cursos.
- Aporta ventajas al alumno y al profesor.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

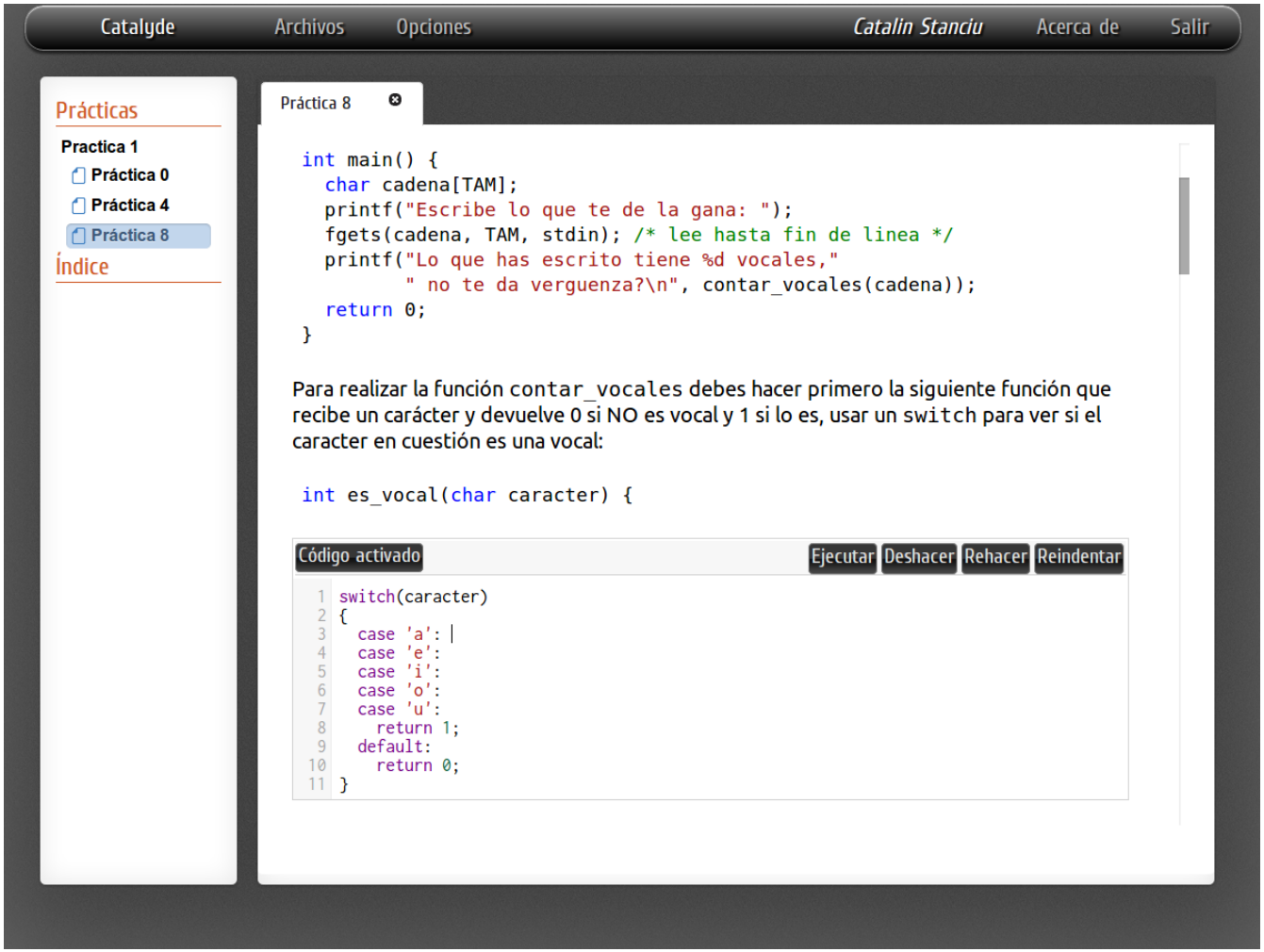
Aportaciones de la aplicación

- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Aportaciones de la aplicación

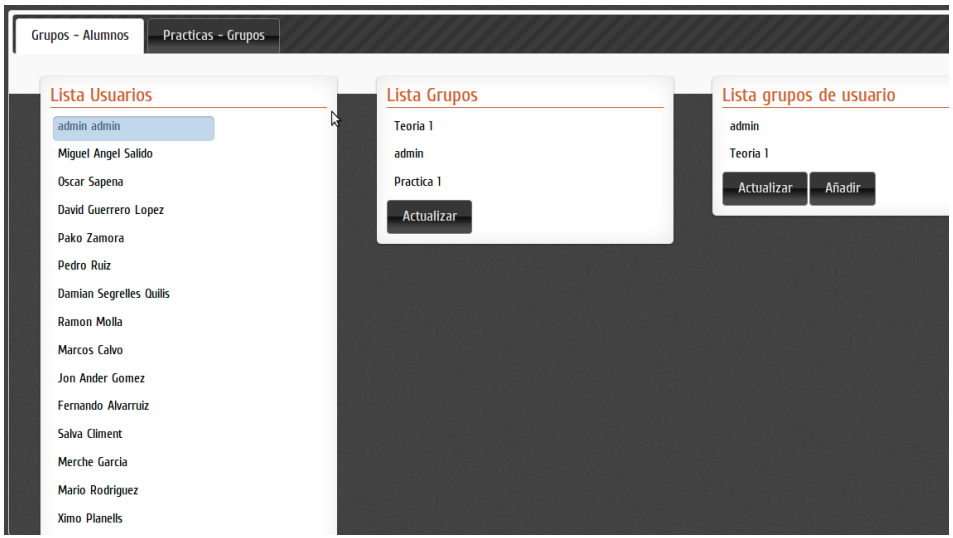
- El alumno puede acceder a las prácticas correspondientes a su grupo y puede editar su código, compilarlo y ejecutarlo.
- Se integran enunciados y ejercicios prácticos en un mismo documento visible por el usuario de la aplicación.
- Es posible activar y desactivar zonas de código de manera incremental, utilizando o no el código del profesor según convenga.
- Posibilidad de realizar tests sobre el código del alumno.
- La infraestructura que se ofrece da la posibilidad de representar visualmente la salida del programa, haciendo más atractivo el desarrollo de los ejercicios prácticos.

Demostración



Vista del profesor

- Sistema de edición de prácticas.
- Interfaz de administración.




```
## Ejercicio sencillo para completar

Completa el siguiente programa que calcula el área de un rectángulo. Para ello debes:

- Escribir una función llamada `area_rectangulo` para que se pueda utilizar en el programa
  la función).
- Completar el programa principal para que pida la altura.

```
#include <stdio.h>
```

<!-- BEGINBLOCK { "b_tag": "funcion_area_rectangulo" , "type" : "code", "checkbutton": "tr
  [{"name" : "Ejecutar", "action" : "run_area" }]}   -->
/* aqui va el codigo de la función area_rectangulo */
<!-- ENDBLOCK -->

```
int main() {
  float base, altura, area;
  fprintf(stdout,"Introduce la base: ");
  fscanf(stdin,"%f",&base);
```

<!-- BEGINBLOCK { "b_tag": "pedir_altura" , "type" : "code", "checkbutton": "true", "edita
  "Ejecutar", "action" : "run_area" }]}   -->
/* aqui va el codigo que pide la altura */
<!-- ENDBLOCK -->

```
  area = area_rectangulo(base,altura);
  fprintf(stdout,"El area es: %f\n",area);
  return 0;
}
```
```

Arquitectura

Cliente

- El cliente se encarga de:
  - Interfaz de usuario.
  - Interacción con el programa ejecutado mediante la terminal.

Servidor

- El servidor se encarga de:
  - Autenticación de usuarios.
  - Gestión de grupos y prácticas.
  - Servir una práctica: mostrar el boletín con los bloques a editar, los botones de configuración y de ejecutar, etc.
  - Realizar las acciones de compilar y ejecutar.

Arquitectura

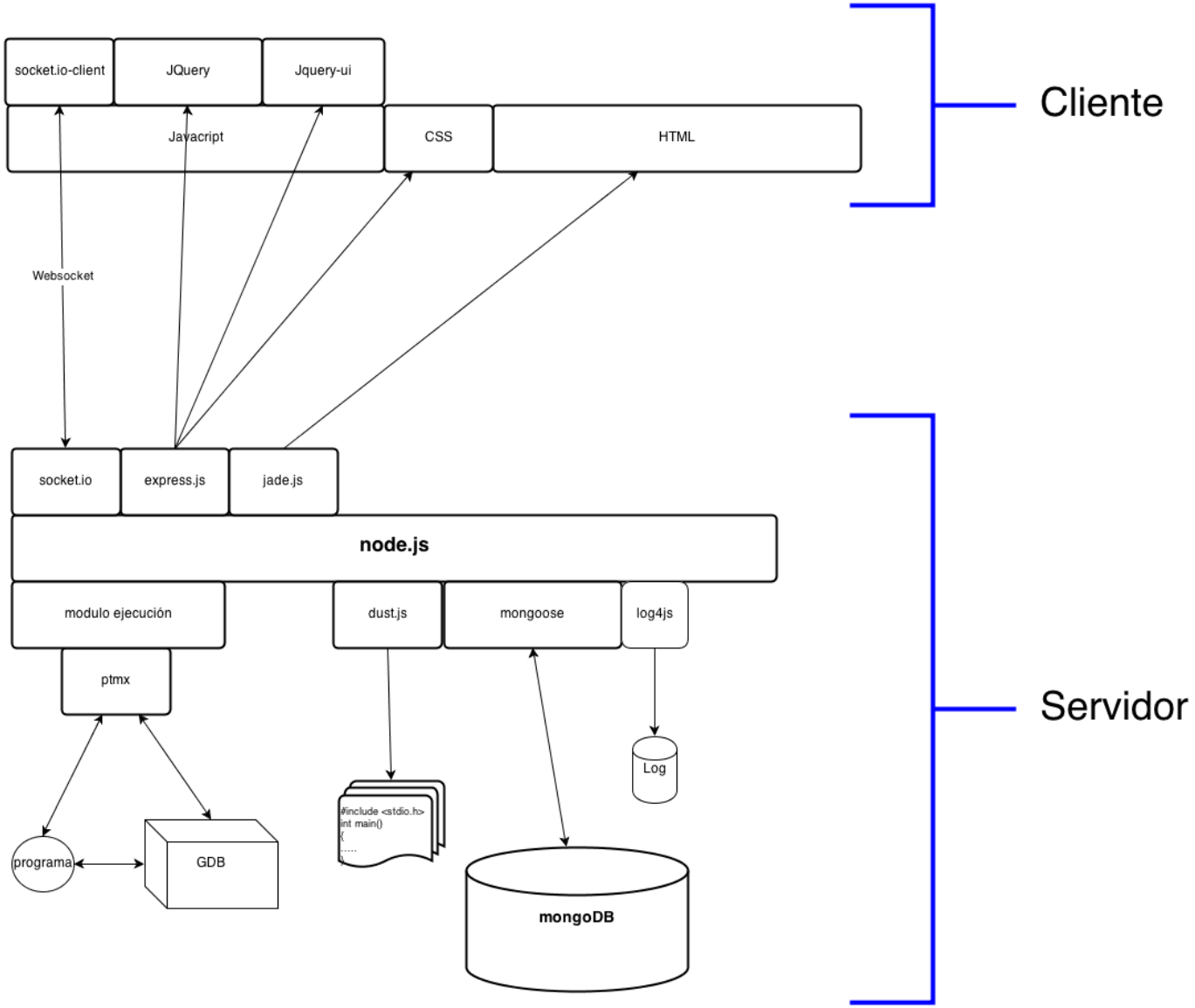
Cliente

- El cliente se encarga de:
  - Interfaz de usuario.
  - Interacción con el programa ejecutado mediante la terminal.

Servidor

- El servidor se encarga de:
  - Autenticación de usuarios.
  - Gestión de grupos y prácticas.
  - Servir una práctica: mostrar el boletín con los bloques a editar, los botones de configuración y de ejecutar, etc.
  - Realizar las acciones de compilar y ejecutar.

Arquitectura



Tecnologías

Tecnologías

Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.

Express

- Convierte nodejs en un servidor web proporcionado:
  - Permisos de acceso
  - Rutas
  - Servidor contenido estático

Mongodb

- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Tecnologías

Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.

Express

- Convierte nodejs en un servidor web proporcionado:
  - Permisos de acceso
  - Rutas
  - Servidor contenido estático

Mongodb

- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Tecnologías

Nodejs

- Librería asíncrona dirigida por eventos.
- Máquina virtual Javascript V8.

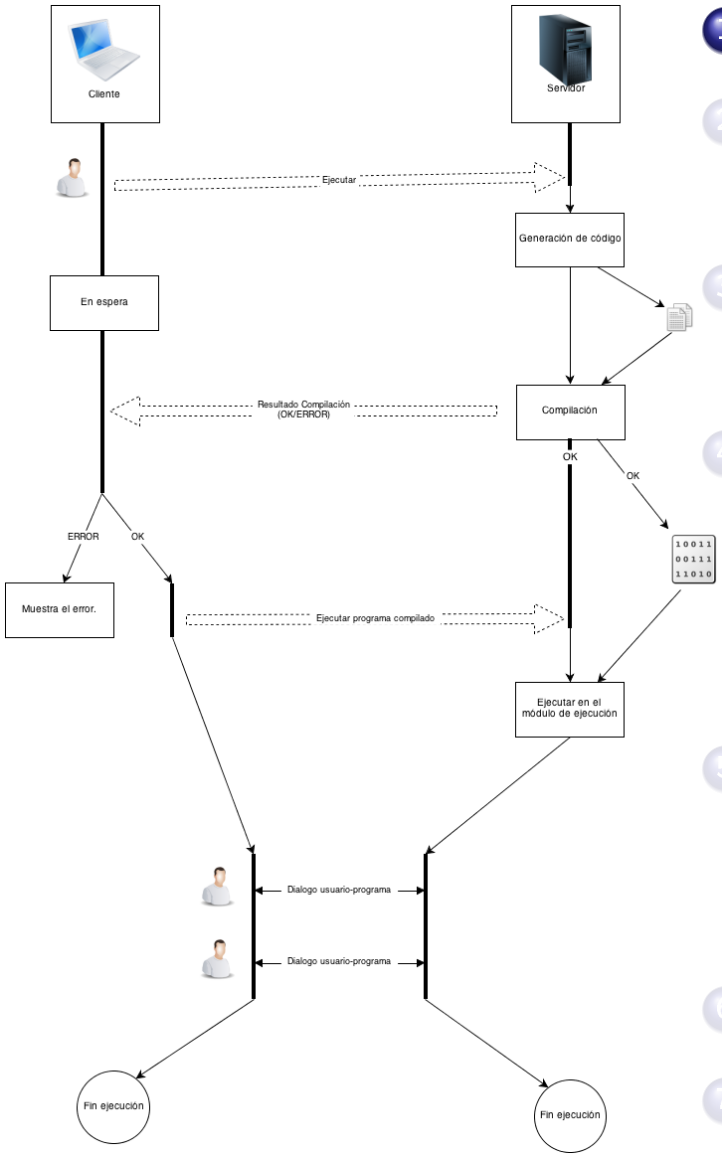
Express

- Convierte nodejs en un servidor web proporcionado:
  - Permisos de acceso
  - Rutas
  - Servidor contenido estático

Mongodb

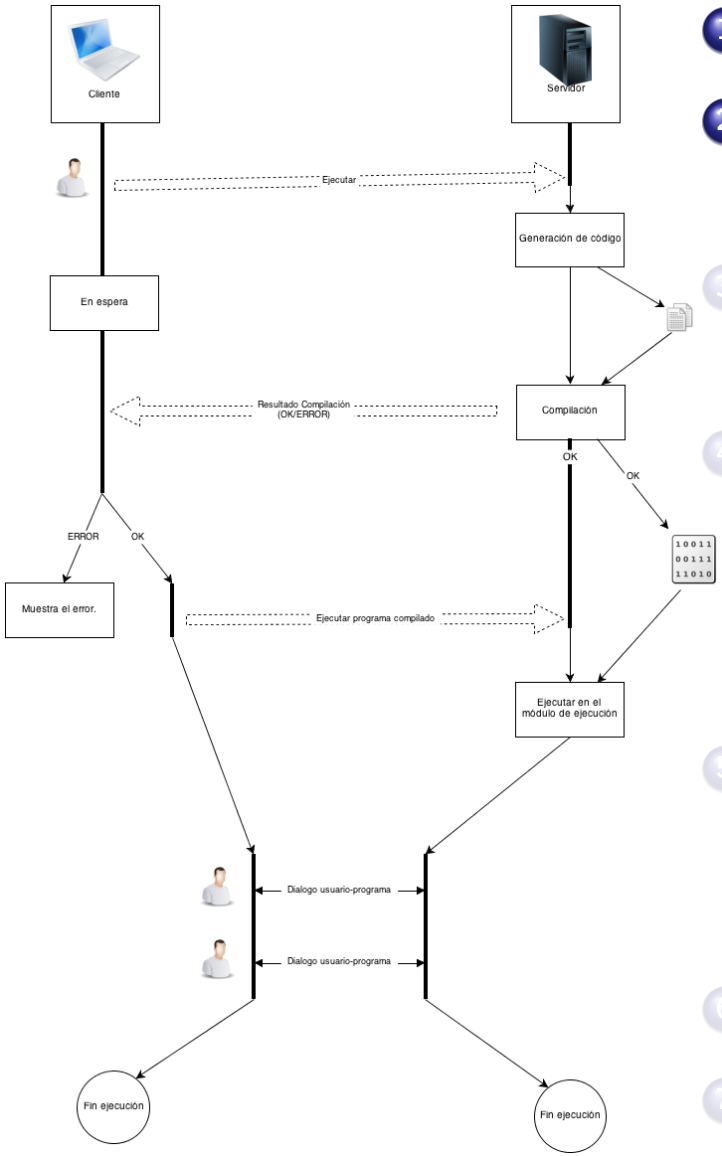
- Base de datos NoSQL
- Rápido acceso a los datos y fácilmente escalable

Compilación y ejecución



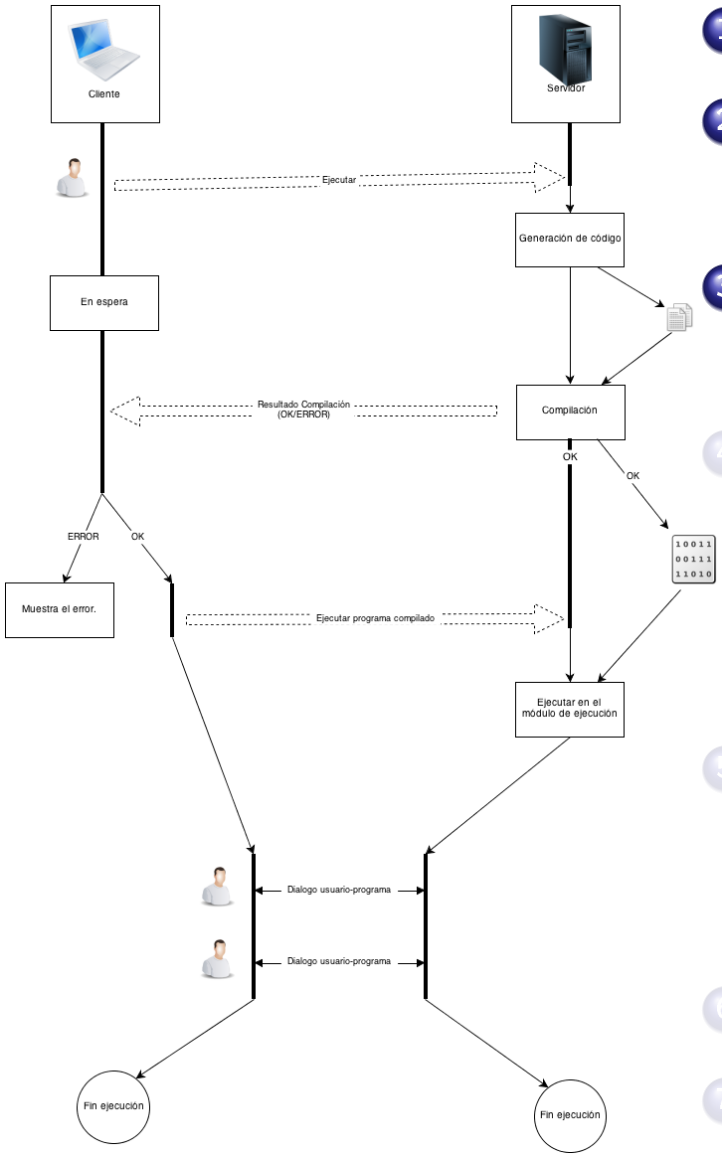
1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



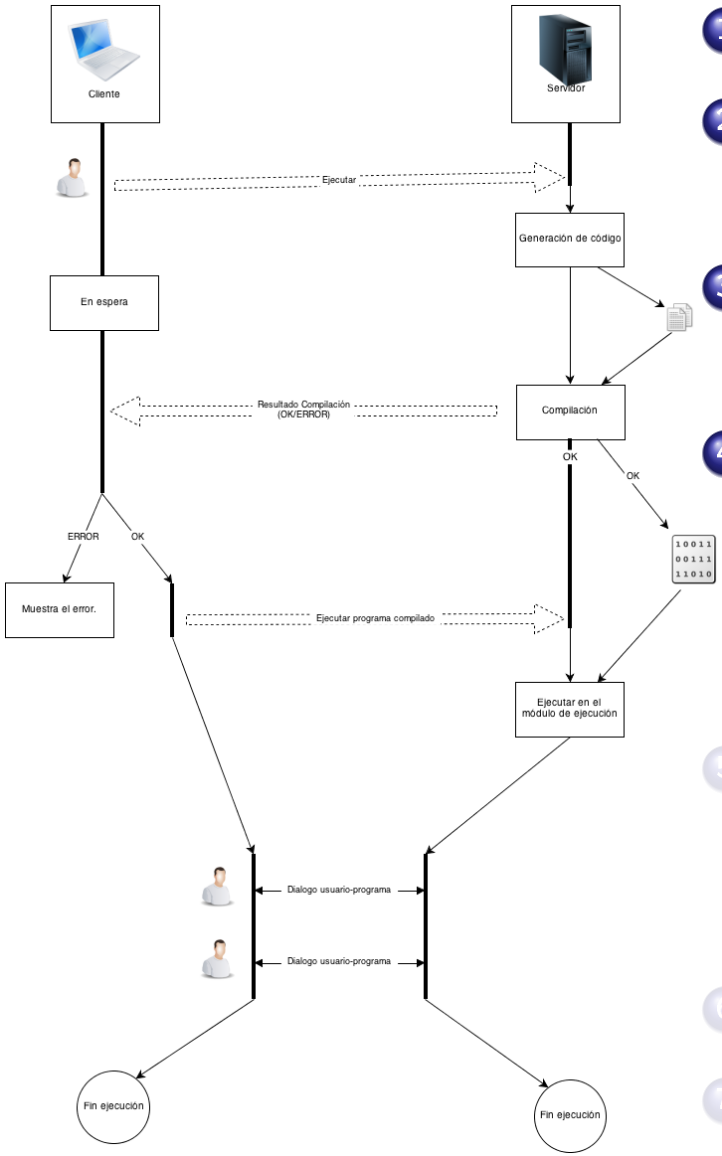
1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



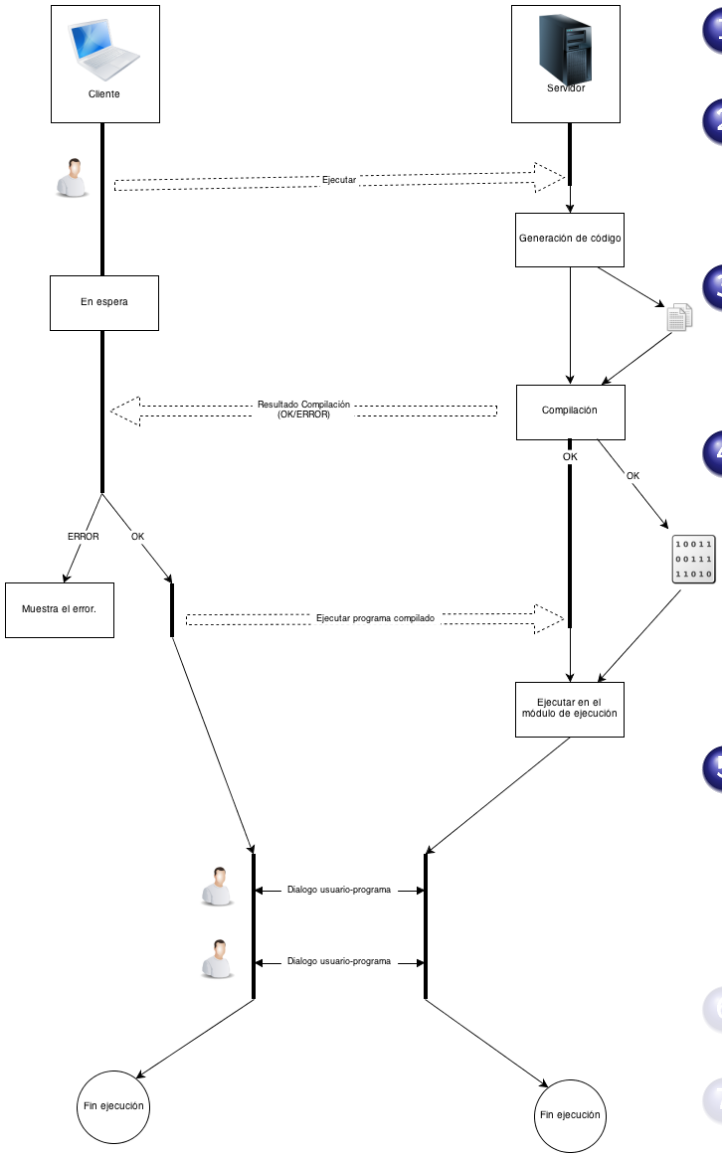
1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



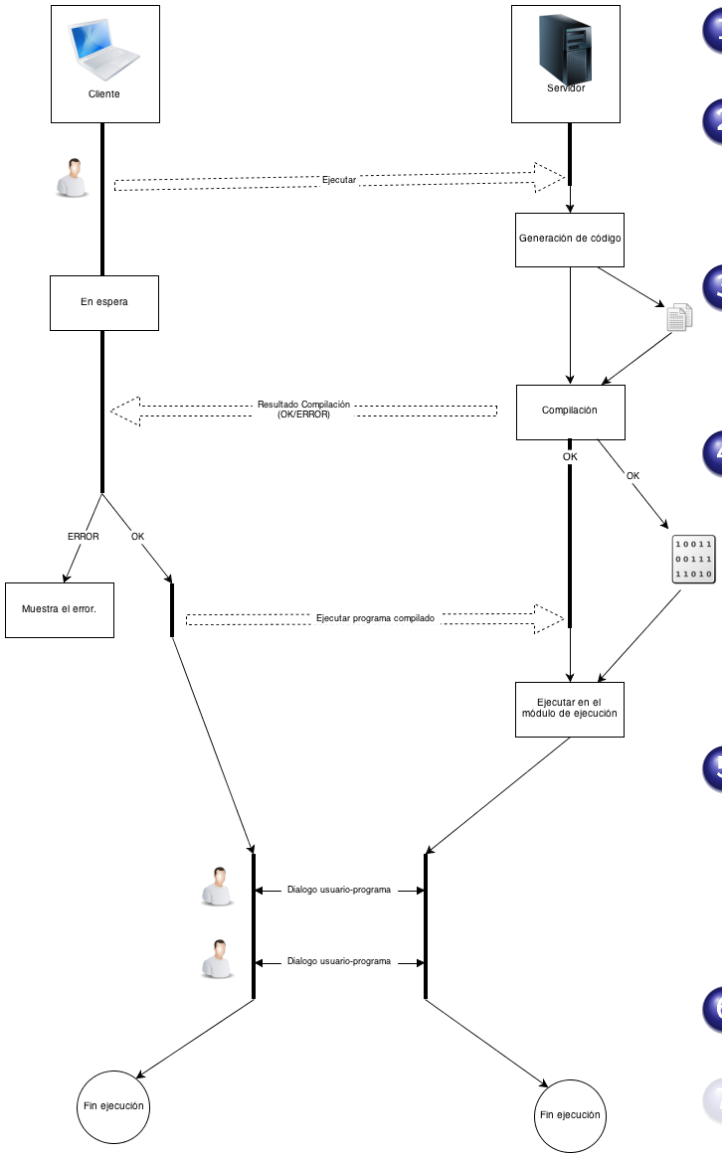
1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



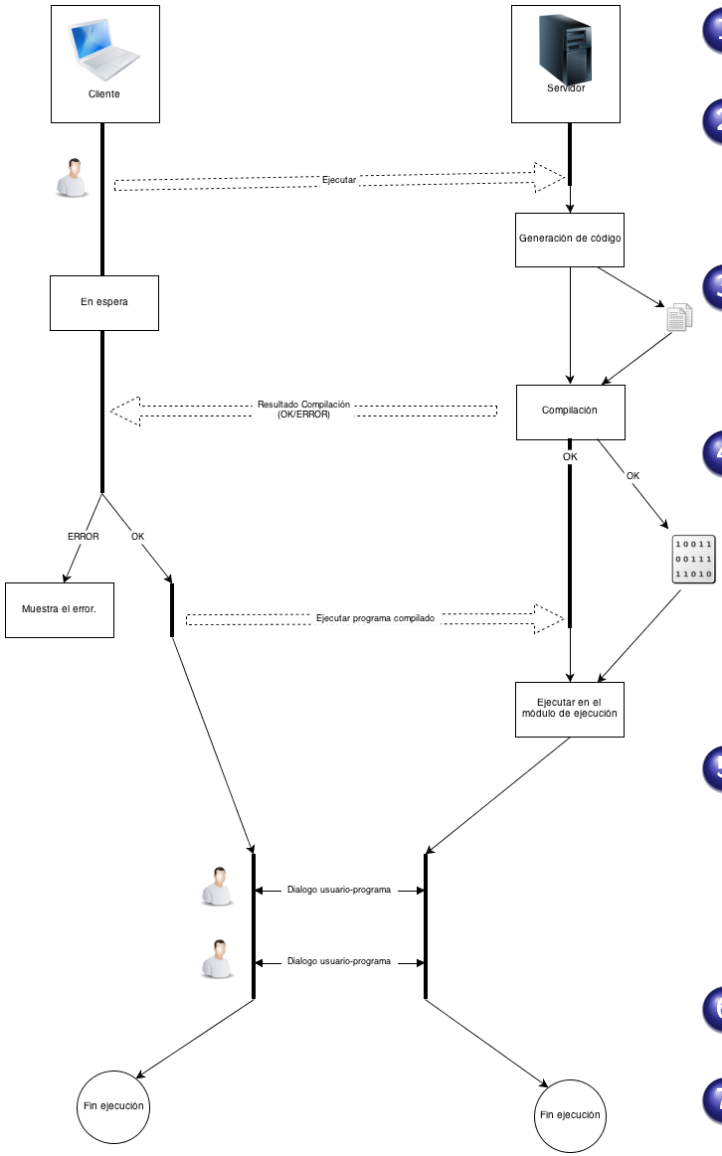
1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Compilación y ejecución



1. El cliente manda el comando de ejecutar.
2. El servidor construye los ficheros fuente y pasa a la fase de compilación.
3. El servidor compila el código y devuelve la respuesta al cliente.
4. El cliente lee la respuesta. En caso de compilarse correctamente se pasa al estado ejecutar, en caso contrario se detiene la operación.
5. Se empieza a ejecutar el programa en el servidor y se comunica la entrada y salida por websocket para permitir interacción.
6. El cliente interacciona con el programa.
7. Se finaliza la ejecución.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Conclusiones

- Obtención de una herramienta enfocada a estudiantes principiantes.
- Desarrollo de una herramienta que tiene aplicación real.
- Integración de tecnologías novedosas.
- Interfaz sencilla y amigable.
- Integración de enunciados y programas.
- Ejecución de los programas en el navegador (multiplataforma).
- Activación y desactivación del código.
- Verificación automática de los programas.

Trabajo futuro

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

Trabajo futuro

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

Trabajo futuro

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

Trabajo futuro

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

Trabajo futuro

- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

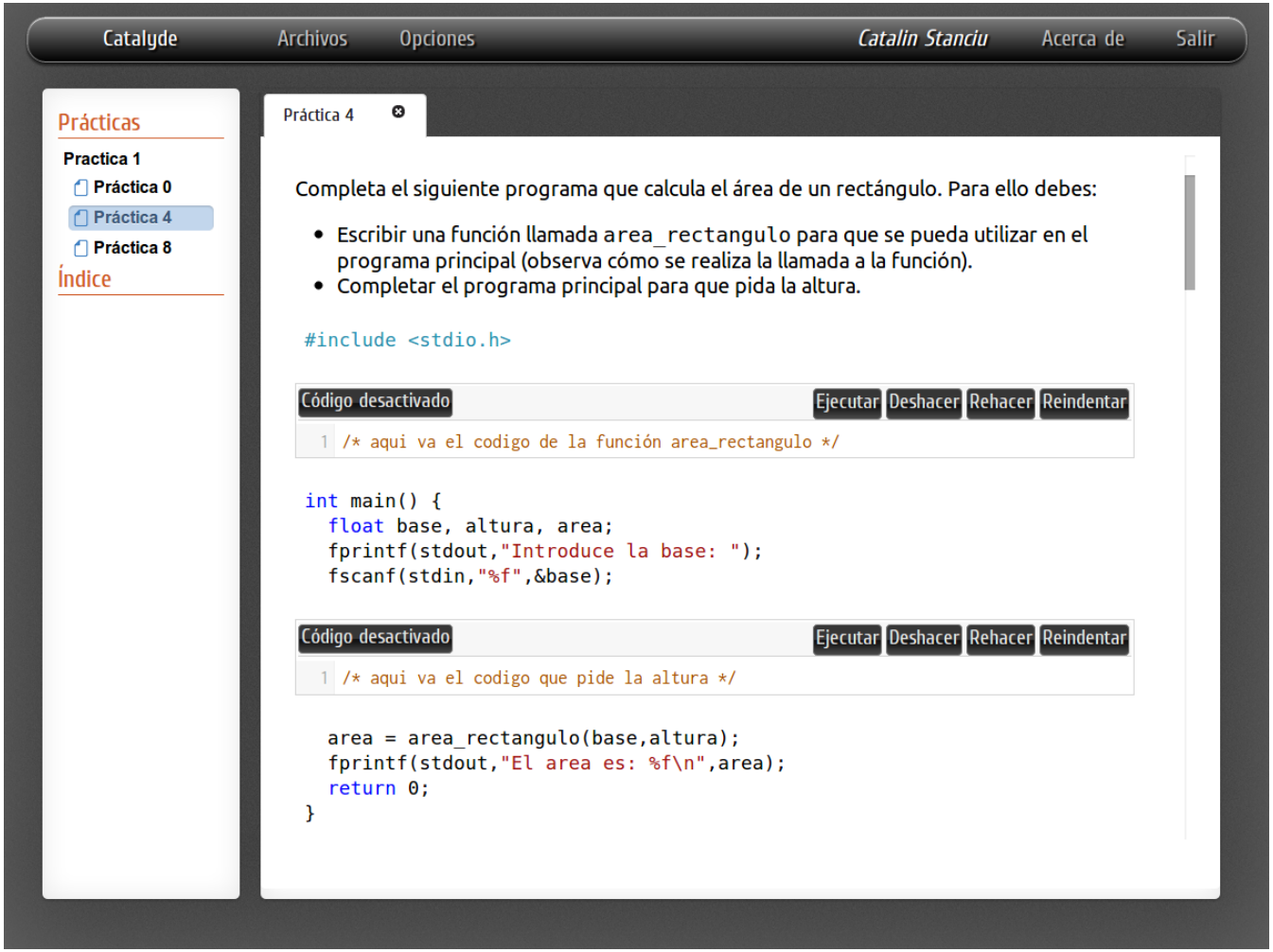
Trabajo futuro

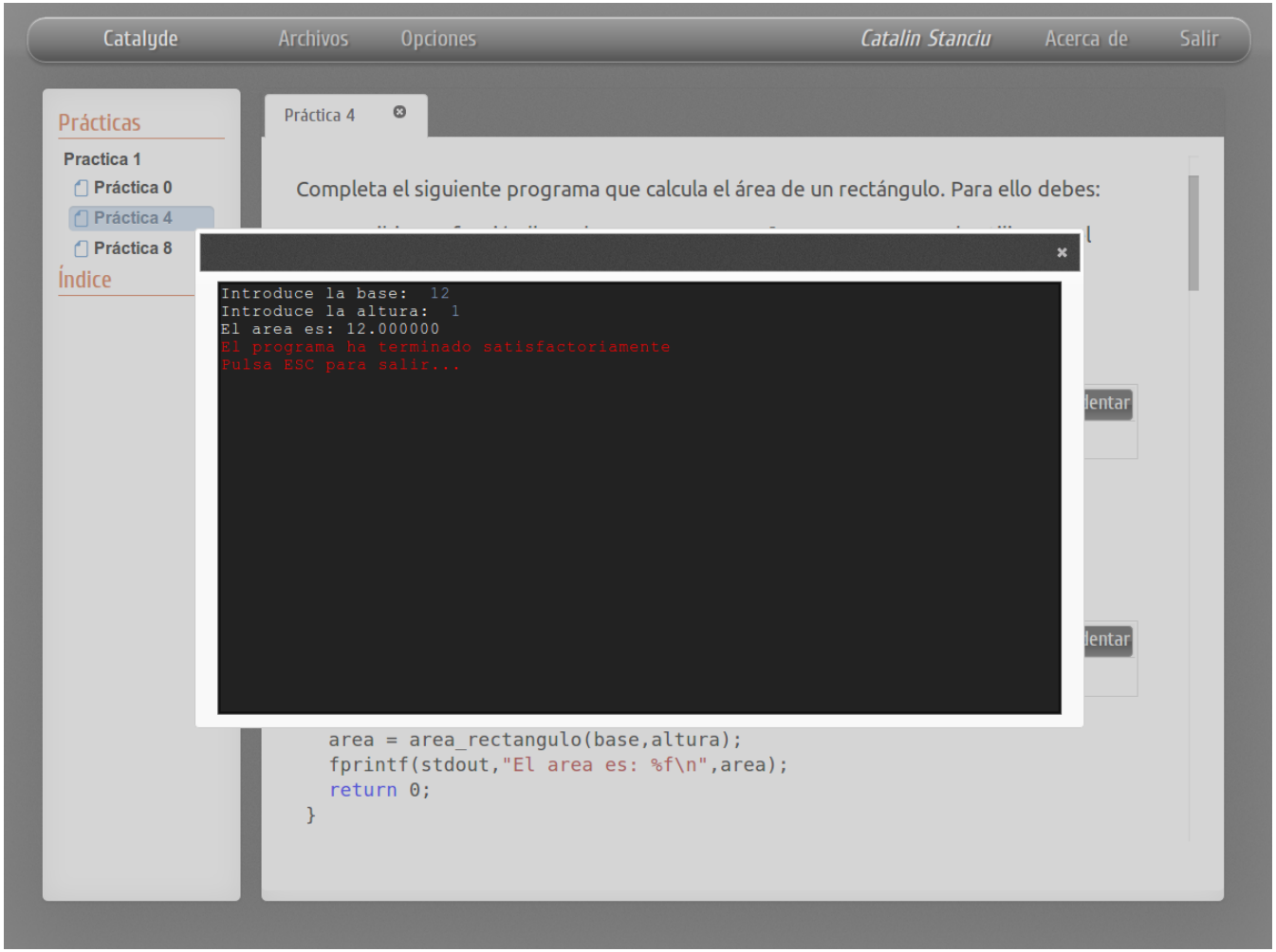
- Mejorar el sistema de creación de las prácticas.
- Trabajo colaborativo.
- Estadísticas y modo de evaluación.
- Ejecución en modo depuración.
- Modo examen.
- Interacción a distancia con el profesor.

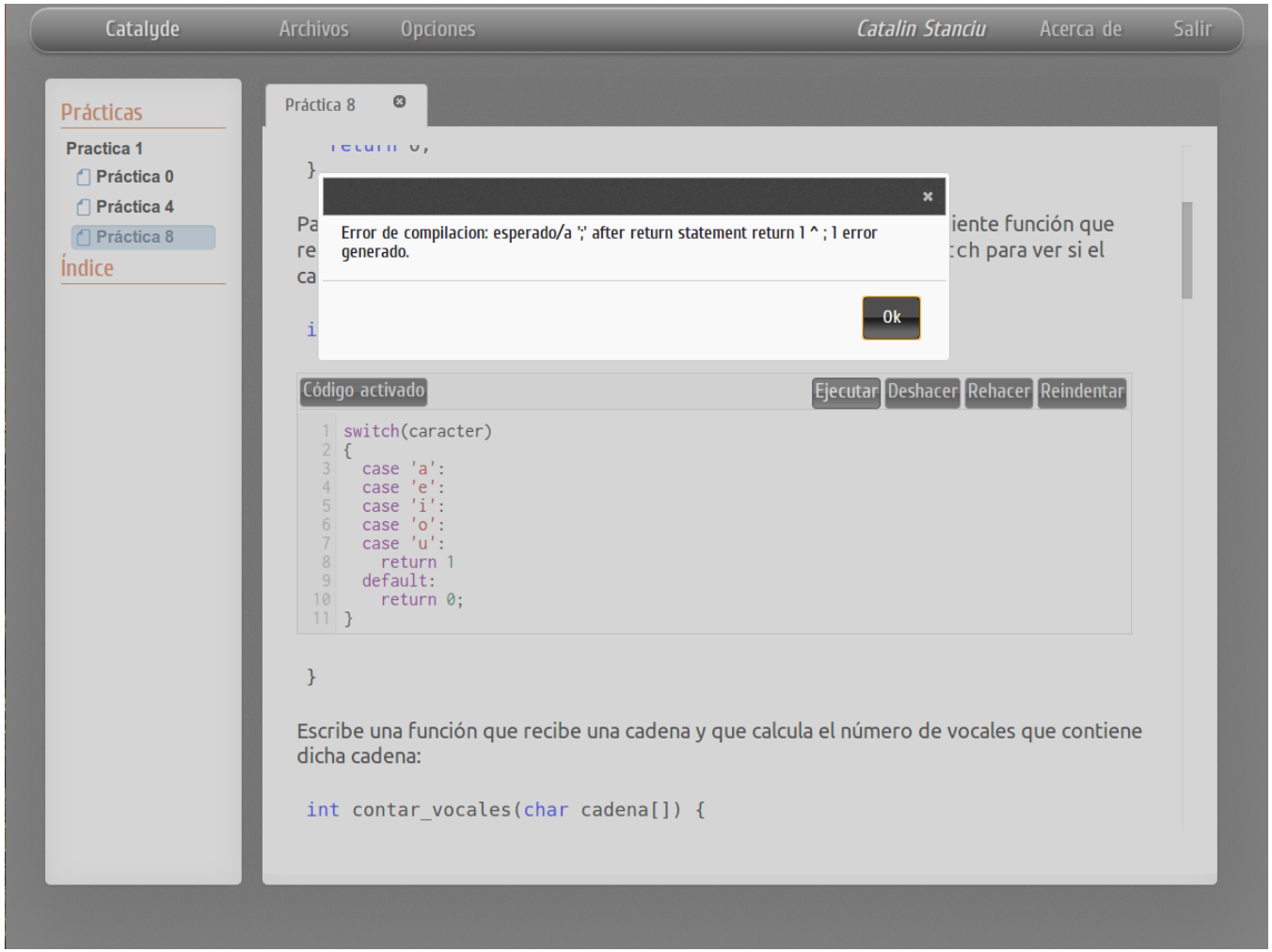
Publicaciones

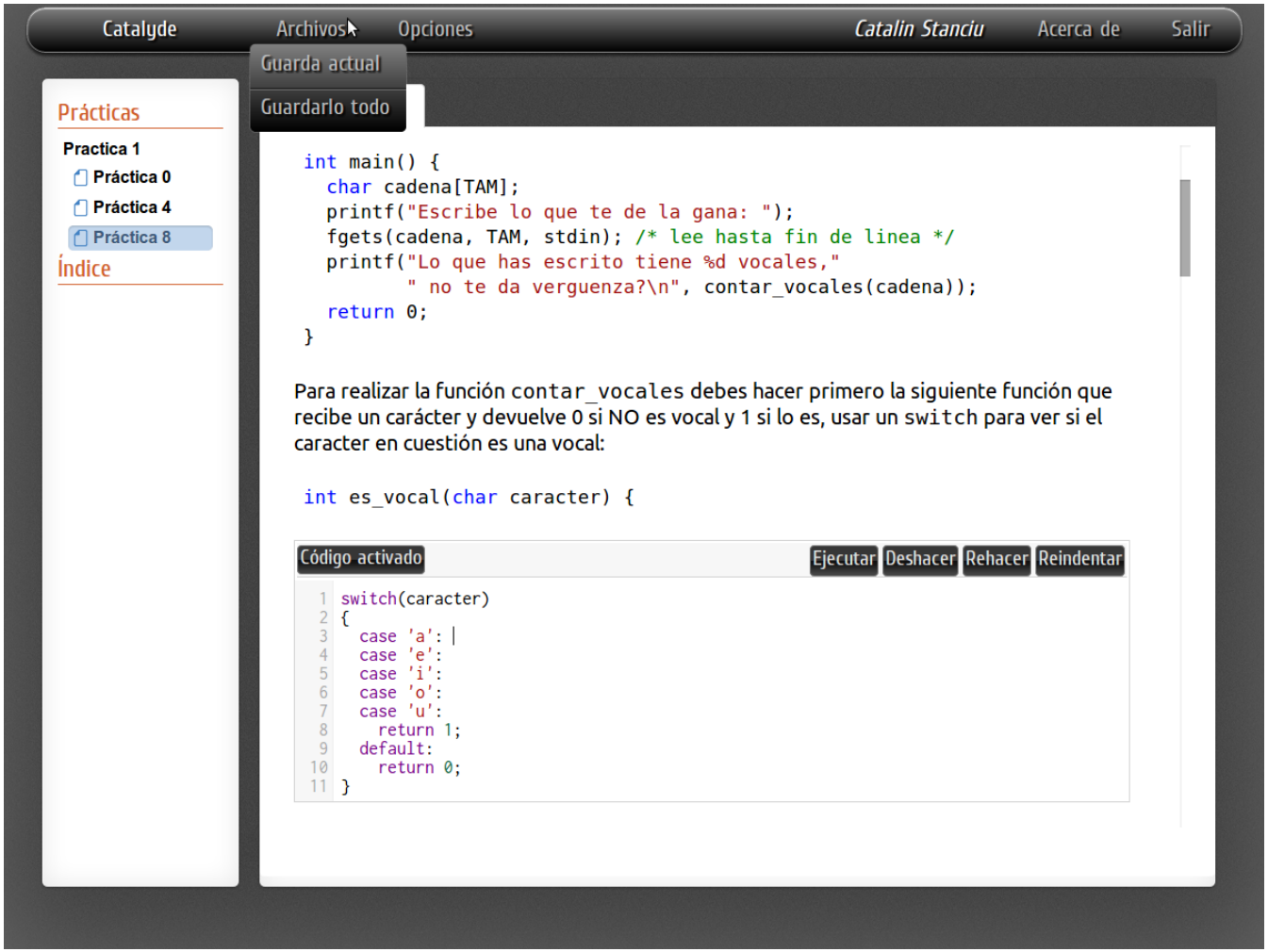
Catalin Costin Stanciu, Joan Pastor-Pellicer, and Salvador España Boquera. Catalyde: herramienta para enseñar programación. In V Jornada de Innovación Docente, Valencia, Spain, December 2012. URL: http://jidinf12.webs.upv.es/poster.

Gracias por su atención.


Catalyde
Archivos
Opciones
Catalin Stanciu
Acerca de
Salir
Prácticas
Practica 1
Práctica 0
Práctica 4
Práctica 8
Índice
Práctica 4
Completa el siguiente programa que calcula el área de un rectángulo. Para ello debes:
Escribir una función llamada area_rectangulo para que se pueda utilizar en el programa principal (observa cómo se realiza la llamada a la función).
Completar el programa principal para que pida la altura.
#include <stdio.h>
Código desactivado
Ejecutar
Deshacer
Rehacer
Reindentar
1 /* aqui va el codigo de la función area_rectangulo */
int main() {
float base, altura, area;
fprintf(stdout,"Introduce la base: ");
fscanf(stdin,"%f",&base);
Código desactivado
Ejecutar
Deshacer
Rehacer
Reindentar
1 /* aqui va el codigo que pide la altura */
area = area_rectangulo(base,altura);
fprintf(stdout,"El area es: %f\n",area);
return 0;
}


Catalyde
Archivos
Opciones
Catalin Stanciu
Acerca de
Salir
Prácticas
Practica 1
Práctica 0
Práctica 4
Práctica 8
Índice
Práctica 4
Completa el siguiente programa que calcula el área de un rectángulo. Para ello debes:
Introduce la base: 12
Introduce la altura: 1
El area es: 12.000000
El programa ha terminado satisfactoriamente
Pulsa ESC para salir...
area = area_rectangulo(base,altura);
fprintf(stdout,"El area es: %f\n",area);
return 0;
}


Catalyde
Archivos
Opciones
Catalin Stanciu
Acerca de
Salir
Prácticas
Practica 1
Práctica 0
Práctica 4
Práctica 8
Índice
Práctica 8
}
Error de compilacion: esperado/a ';' after return statement return 1 ^ ; 1 error generado.
Ok
Código activado
Ejecutar
Deshacer
Rehacer
Reindentar
1 switch(caracter)
2 {
3 case 'a':
4 case 'e':
5 case 'i':
6 case 'o':
7 case 'u':
8 return 1
9 default:
10 return 0;
11 }
}
Escribe una función que recibe una cadena y que calcula el número de vocales que contiene dicha cadena:
int contar_vocales(char cadena[]) {


Catalyde
Archivos
Opciones
Catalin Stanciu
Acerca de
Salir
Guarda actual
Guardarlo todo
Prácticas
Practica 1
Práctica 0
Práctica 4
Práctica 8
Índice
int main() {
char cadena[TAM];
printf("Escribe lo que te de la gana: ");
fgets(cadena, TAM, stdin); /* lee hasta fin de linea */
printf("Lo que has escrito tiene %d vocales,"
" no te da verguenza?\n", contar_vocales(cadena));
return 0;
}
Para realizar la función contar_vocales debes hacer primero la siguiente función que recibe un carácter y devuelve 0 si NO es vocal y 1 si lo es, usar un switch para ver si el caracter en cuestión es una vocal:
int es_vocal(char caracter) {
Código activado
Ejecutar
Deshacer
Rehacer
Reindentar
switch(caracter)
{
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
return 1;
default:
return 0;
}