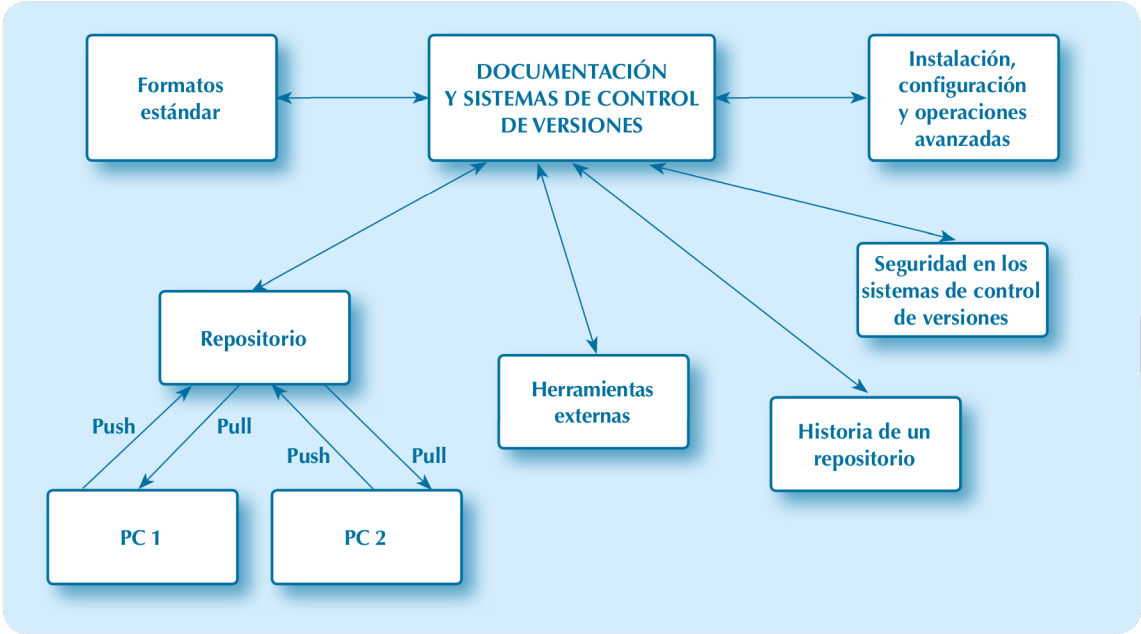
6

Documentación y sistemas de control de versiones

Objetivos

- ✓ Identificar las diferentes herramientas de generación de documentación.
- ✓ Documentar los componentes software utilizando los generadores específicos de las plataformas.
- ✓ Usar diferentes formatos para la documentación.
- ✓ Utilizar herramientas colaborativas para la elaboración y mantenimiento de la documentación.
- ✓ Instalar, configurar y usar un sistema de control de versiones.
- ✓ Garantizar la accesibilidad y seguridad de la documentación almacenada por el sistema de control de versiones.

Mapa conceptual



Glosario

Array. Se le llama también *vector*. Es una colección de elementos del mismo tipo, y existen varias clases como unidimensionales, bidimensionales, etc.

Bug. Es un error de software que provoca un resultado no deseado. También puede ser un problema de seguridad que permite un ataque al software por parte de personal externo.

Historia de un repositorio. Es un listado de cualquier repositorio que permite observar los diferentes commits, push, creación de ramas del ciclo de vida del desarrollo del proyecto. Cada cambio en las diferentes ramas permite generar una versión del software. Todos los sistemas de control de versiones deben contenerlo.

IDE *(Integrated Development Environment)*. Es un programa informático que permite al programador desarrollar todo el software relacionado con un proyecto o aplicación web.

Plantilla. Es un texto o código que puede ser usado varias veces y permite ahorrar tiempo a la hora de implementar cualquier elemento en programación.

Tags. Son unas etiquetas que se aplican en el entorno de programación, por ejemplo, cuando se documenta un proyecto software.

Template. También se le llama *plantilla*. Es un documento base, a partir del cual se puede programar o generar documentación.

Versión. Indica numéricamente cuál es el avance en el desarrollo de una aplicación software.

6.1. Introducción

En este último capítulo se va a tratar la documentación de una aplicación web, tan fundamental como el código de la misma. Por otro lado, también se tratará el concepto de control de versiones de una aplicación cuando se está desarrollando por parte de un equipo de programadores. Su objetivo final es poner en producción la aplicación web para que se use con un objetivo específico. Y es esencial que el equipo de trabajo suba las distintas versiones probadas de la aplicación; lo ideal sería modular, y que cada programador se encargue de un módulo.

6.2. Herramientas externas para la generación de documentación. Instalación, configuración y uso

Antes de comenzar a explicar o entrar en detalle en herramientas para generar documentación, sería necesario tener en cuenta qué componentes o módulos serían necesarios documentar. Normalmente es un equipo de trabajo el que lleva a cabo la programación de una aplicación web, dependiendo de la dimensión del proyecto. Por ello, es preciso documentar toda aquella parte del código que será reutilizable o que pueda ser modificada por una versión posterior o por algún error de flujo de datos.

En el mercado actual existen bastantes herramientas para la generación de documentación, pero sería imposible tratar todas ellas en este libro; se va a realizar un resumen de algunas de ellas y se instalarán algunas para observar cómo se comportan. La lista de las herramientas que generan automáticamente documentación son las siguientes:

- *Javadoc:* es la herramienta de Oracle por excelencia para generar documentación en formato HTML a partir del código desarrollado en Java. Hay que explicarla, ya que se han puesto bastantes ejemplos a lo largo del libro. La mayoría de los IDE de Java generan automáticamente la documentación, y uno de ellos es Netbeans 8.2, pero todas las versiones tienen esta posibilidad.
- *phpDocumentor:* es otra herramienta escrita en PHP para generar documentación automáticamente vía código fuente PHP y como salida tiene el estándar PHPDoc.
- *Doxygen:* esta herramienta es más versáti,l ya que permite generar documentación en bastantes lenguajes de programación, como por ejemplo C++, Java, C, Fortran, VDHL, CLI, Python, IDL y también con ciertos matices PHP y D. Funciona en la mayoría de los sistemas Unix, Windows y MAC OS X.

Actividad propuesta 6.1

Investiga en Internet qué herramientas existen en el mercado, además de las nombradas anteriormente, e intenta instalar y configurar alguna de ellas.

6.2.1. Instalación, configuración y uso de Javadoc

La instalación de Javadoc no es necesaria en Windows, puesto que viene incorporado el componente en el IDE Netbeans, por lo que se puede configurar y usar directamente sobre el código Java desarrollado en cualquier proyecto.

A partir de un proyecto generado de cualquier aplicación se pueden documentar las clases, los métodos, las funciones, etc. Para escenificar este uso, se pondrá el siguiente ejemplo de un proyecto de Java relacionado con XML:

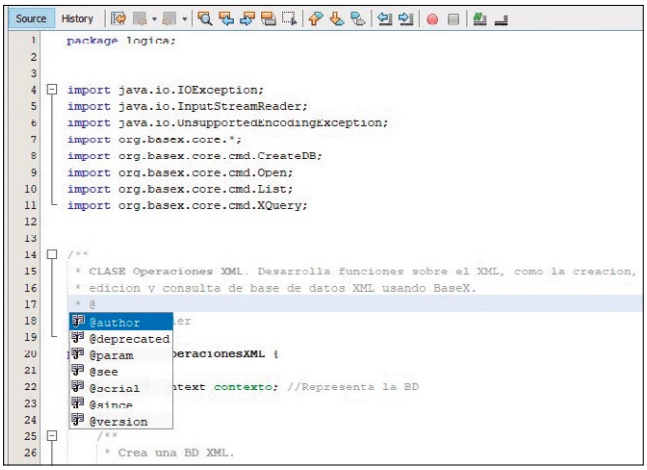


Figura 6.1
Proyecto Java

Si se observa la imagen, es un código de una clase de Java con sus funciones. Lo normal en una clase es documentar el nombre, una descripción general, la versión y el nombre de autor o autores. Si se documenta la clase, se podrán documentar los siguientes parámetros:

- ✓ *@author:* este atributo permite declarar el nombre del autor de la clase.
- ✓ *@deprecated:* para declarar si algún elemento (por ejemplo, la clase) está obsoleto y no se recomienda su uso. Puede ser por varias razones, una de ellas es porque existe un problema de seguridad o bug, otra sería porque la forma de programar esa parte son malas praxis, y la última porque existe una versión más actual.
- ✓ *@param:* definición de un parámetro de un método.
- ✓ *@see:* se asocia con otro método o clase.
- ✓ *@serial:* describe el motivo del campo y sus posibles valores.
- ✓ *@since:* la versión del producto que se está desarrollando.
- ✓ *@version:* es la versión numérica de una clase o un método. Se puede comenzar por 0 o por 1, y, posteriormente, ir incrementando a medida que se implementen más versiones, por ejemplo 1.1. La secuencia la determina el programador o el equipo de programación.

TOMA NOTA

Es importante introducir la @ en la documentación con el parámetro correspondiente, ya que, en caso contrario, Javadoc dará un error por no poder generar la documentación adecuada.

Para documentar un método se visualizan algunos parámetros similares y otros distintos, como se observa en la figura 6.2.

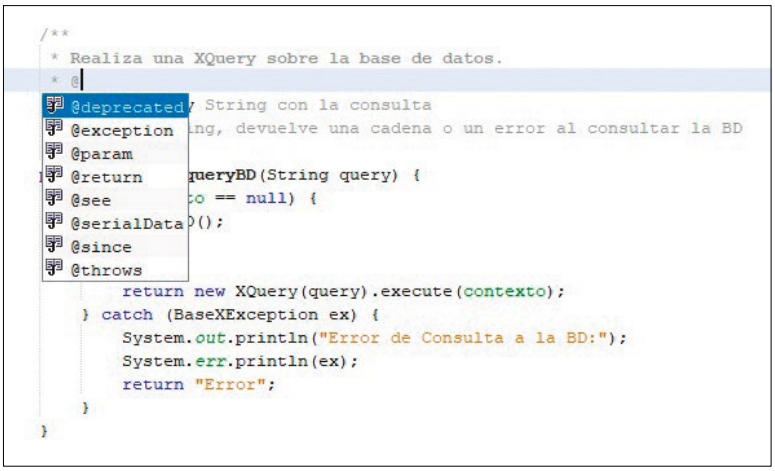


Figura 6.2
Documentar método

Los parámetros del método son los siguientes:

- ✓ *@deprecated:* para declarar si algún elemento del método está obsoleto y no se recomienda su uso. Puede ser por varias razones, una de ellas es porque existe un problema de seguridad o bug, otra sería porque la forma de programar esa parte son malas praxis, y la última porque existe una versión más actual.
- ✓ *@exception:* es sinónimo de *@throws,* se inventó primero. Con el paso del tiempo los expertos concluyeron que era más exacto @throws.
- ✓ *@param:* describe el parámetro o parámetros que recibe el método.
- ✓ *@return:* describe el valor de salida del método; si el método es void no devuelve nada.
- ✓ *@see:* se asocia con otro método o clase.
- ✓ *@serialData:* describe el formato de serialización usado en el método.
- ✓ *@since:* la versión del producto que se está desarrollando.
- ✓ *@throws:* describe una excepción lanzada por el método que debe ser tenida en cuenta.

Una vez documentada la clase y los métodos de la misma se está en disposición de generar la documentación. Para ello, se ejecuta Javadoc dentro de la opción *Ejecutar* y pulsando en *Generar Javadoc* en el IDE Netbeans, como se observa en la figura 6.3.

Si se pulsa la opción anterior se visualizará una página como la siguiente, donde aparecerá toda la documentación generada por Javadoc de tu aplicación (figura 6.4)

Actividad propuesta 6.2

Toma una aplicación como ejemplo y genera documentación con Javadoc, previamente se deben documentar las clases y sus métodos.

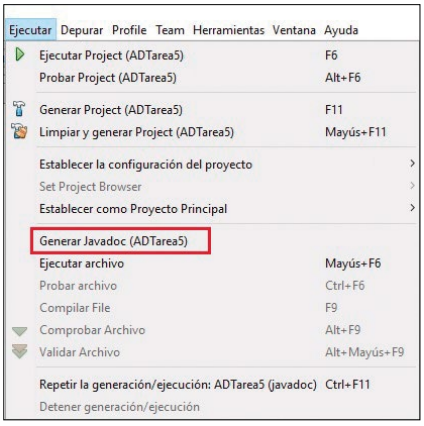


Figura 6.3
Javadoc

Figura 6.4
Salida Javadoc

En esta salida se pueden observar las clases realizadas, como son Principal, OperacionesXML y OperacionesXQuery, los métodos con sus parámetros, etc. En conclusión, una perspectiva bastante acertada de la aplicación para observar cuál es su objetivo, cómo está implementada, la relación entre las clases. Por ello es tan importante realizar una documentación de una aplicación desarrollada.

6.2.2. Instalación, configuración y uso de phpDocumentor

Como se ha comentado anteriormente, existen varias herramientas para generar documentación de forma automática y una de ellas es phpDocumentor. Esta herramienta genera documentación para el lenguaje de programación PHP.

Los elementos que pueden ser documentados son los siguientes:

- Variables globales.
- Clases.
- Funciones.
- Métodos y atributos.
- Sentencias.

Por otro lado, se puede documentar también un bloque de código y puede hacer referencia a un archivo en particular, ya que normalmente en PHP la aplicación puede estar compuesta por varios archivos. La etiqueta que permite esta documentación es aquella que se denomina @package. Existen tres tipos de documentaciones relacionadas con el Modelo-Vista-Controlador:

a) *Vista o Interfaz:* qué permite realizar, cómo lo hace, qué retorna, etc.
b) *Modelo:* qué algoritmos usa, qué estructura tiene, qué flujo usa, etc.
c) *Controlador:* qué métodos usa para gestionar el flujo entre la vista y el modelo, cómo optimizar tales métodos, etc.

Recuerda

✓ La documentación de cualquier aplicación es básica, la cual reside dentro del código para que el desarrollador, con un simple vistazo, conozca de primera mano cada función, clase, etc. Luego un documento externo servirá para que cualquier consultor o jefe de proyecto pueda observar claramente la estructura y el funcionamiento de la aplicación a grandes rasgos.

Al igual que en Javadoc, en phpDocumentor existen etiquetas para documentar los bloques de documentación, todas precedidas por la @. Son las siguientes:

✓ *@authtor:* autor que implementa el código.
✓ *@copyright:* derechos de autor.
✓ *@access:* esta marca puede ser privada o pública. Si es privada no genera documentación y si es pública sí. El valor por defecto es pública.
✓ *@deprecated:* indica que un elemento está obsoleto y en futuras versiones del código no debería usarse.
✓ *@internal:* permite indicar la documentación para los programadores, pero no es pública.
✓ *@version:* la versión actual del código.

Y, para las funciones, serían las siguientes:

✓ *@global:* indica el uso global de una variable.
✓ *@return:* valor devuelto por la función.
✓ *@param:* parámetro o parámetros que recibe una función.
✓ *@var:* se usa para documentar los atributos de la clase.

Para saber más

Si quieres profundizar más en phpDocumentor, puedes consultar su página web: https://www.phpdoc.org/

Si se toma una aplicación con una clase o una jerarquía de clases en PHP con sus métodos correspondientes, se puede documentar mediante los tags anteriores y una configuración que es necesario realizar en Netbeans 8.2. La configuración es la siguiente:

1. Se crea un proyecto en PHP mediante el interfaz de Netbeans y se crean los respectivos ficheros que va a contener la aplicación. Es necesario tener instalado XAMPP y Netbeans en el equipo.
2. En la interfaz de Netbeans 8.2 se selecciona la opción *Herramientas* y, dentro de esta, la opción *Opciones*. Se visualizará una pantalla como la 6.5.

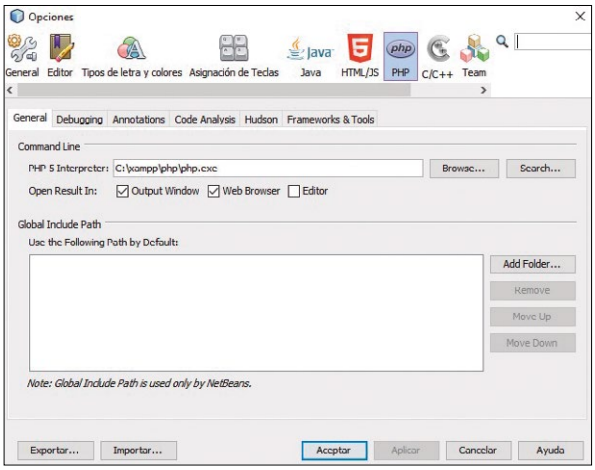


Figura 6.5
Configuración PHP

TOMA NOTA

Puede ser necesario activar la opción PHP en el menú de la figura 6.5, y se visualizarán todas las opciones.

3. Como se observa en la imagen anterior, es necesario colocar dónde está la ruta del intérprete de PHP, que en este caso coincide con la instalación de XAMPP.

   En un paso posterior, es necesario descargarse la versión de phpDocumentor 2.9.0 que es estable y funciona perfectamente. Las siguientes versiones tienen algunos bugs que está solucionando la comunidad de programadores. Para descargar la versión se encuentra disponible en la URL https://www.codingfix.com/es/installing-phpdocumentor-issues-solved/.
4. Una vez descargada, se descomprime en el directorio de XAMPP y se configura en Netbeans en la misma ventana anterior, pero esta vez en la opción Framework & Tools (figura 6.6).
5. El siguiente paso sería configurar las propiedades del proyecto. Para ello es necesario seleccionar el proyecto con el botón derecho del ratón y después seleccionar *Propiedades.* Se visualizará una pantalla: en la parte izquierda es necesario seleccionar *Documentación* y en la parte derecha *Documentation Provider phpDocumentor,* y configurar en directorio de salida de la documentación. Normalmente se crea un directorio llamado doc que cuelga del directorio raíz (figura 6.7).

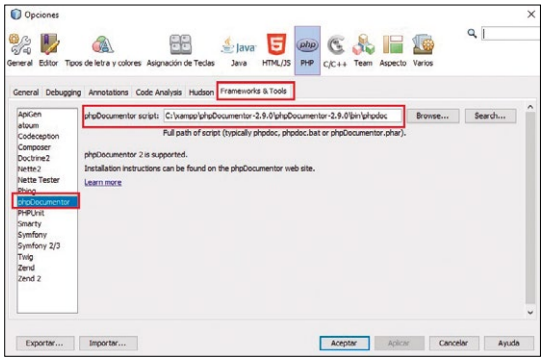


Figura 6.6
Configuración phpDocumentor

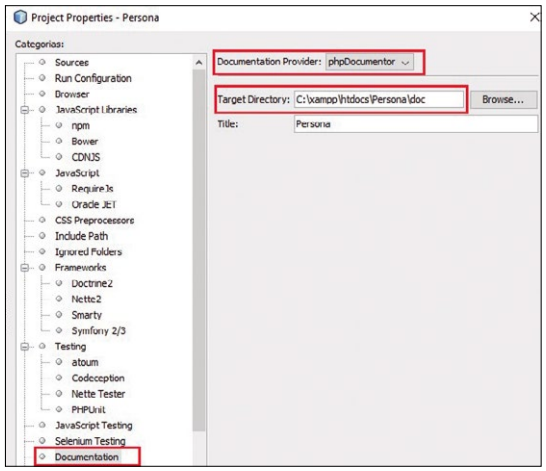


Figura 6.7
Documentación de la aplicación

6. Con este último paso ya se está en disposición de generar la documentación de la aplicación. Para ello se pulsará con el botón derecho del ratón en la opción *Generar documentación* o *Generate Documentation*. Se generará una página HTML como la de la figura 6.8.

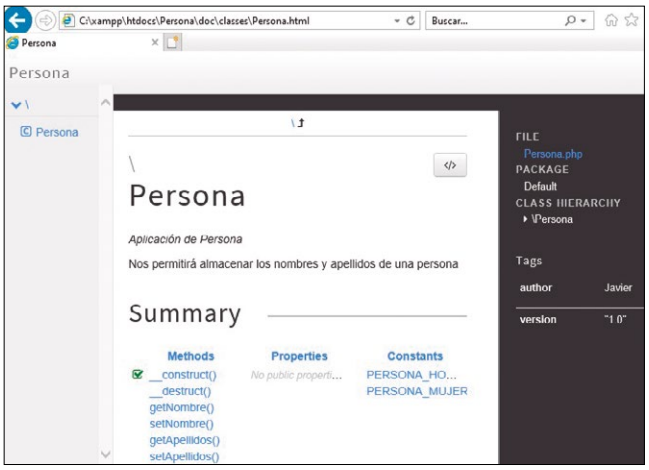


Figura 6.8
Documentación final

Actividad propuesta 6.3

Toma una aplicación como ejemplo y genera documentación con phpDocumentor, previamente se deben documentar las clases y sus métodos. Configurar también phpDocumentor en Netbeans 8.2.

6.2.3. Instalación, configuración y uso de Doxygen

Como se comentó anteriormente, Doxygen es una de las herramientas más versátiles dentro de la generación de documentación, porque permite documentar un abanico amplio de lenguajes de programación. En nuestro caso, se va a tomar como ejemplo una de las prácticas de los módulos que se imparten en informática.

Primero de todo, es necesario descargar Doxygen desde la URL https://www.doxygen.nl/download.html. En tal página está disponible la última versión, que es la número 1.8.20 del 24 de agosto del 2020. Y están disponibles los binarios para Linux, Windows y Mac OS, además del código fuente. La forma de trabajar de Doxygen es en forma de asistente o experto, en nuestro caso, se va a trabajar de forma wizard para hacer más sencilla su comprensión.

Una vez descargada la aplicación en Windows, sería necesario instalarla de una forma fácil e intuitiva. Y, si se ejecuta, se visualizará la interfaz de la figura 6.9.

La aplicación web es una colección de libros con su autor realizada en Java con sus clases y métodos. La configuración de la primera pantalla de Doxygen consistiría en los siguientes ítems. Se observará en la pantalla de la figura 6.10.

- *Project name:* nombre del proyecto.
- *Project synopsis:* breve descripción del proyecto.
- *Project version:* la versión de la aplicación.
- *Project Logo:* una imagen que represente el proyecto.
- *Source Code directory:* el directorio raíz de la aplicación.
- *Destination directory:* el directorio donde se almacenará la documentación generada.

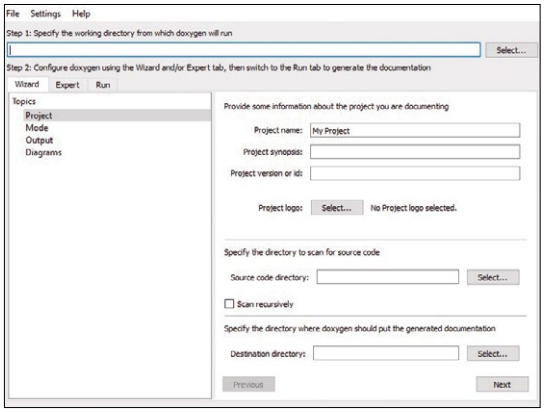


Figura 6.9
Doxygen

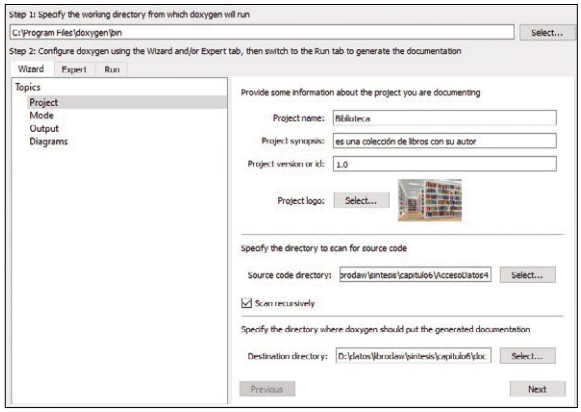


Figura 6.10
Wizard Doxygen

Una vez completada la ventana anterior, se pulsa en *Next* y se visualizarán las siguientes opciones para configurar que corresponden al Mode:

- ✓ *Extraction mode:* se va a seleccionar *All Entities.*
- ✓ *Select programming language:* en nuestro caso es *Java.*

Si se pulsa *Next* se llegarán a las siguientes opciones, como se puede observar en la figura 6.11:

- *Output format to generate:* en nuestro caso, se selecciona HTML, y posee la opción de texto plano o navegación mediante panel. También permite la opción de cambiar el color de la documentación HTML.
- *Latex:* posee distintas opciones, se puede seleccionar como formato intermedio para enlazarlo con pdf.
- *Opciones de ficheros:* man pages, RTF o XML. Se seleccionan páginas de ayuda.

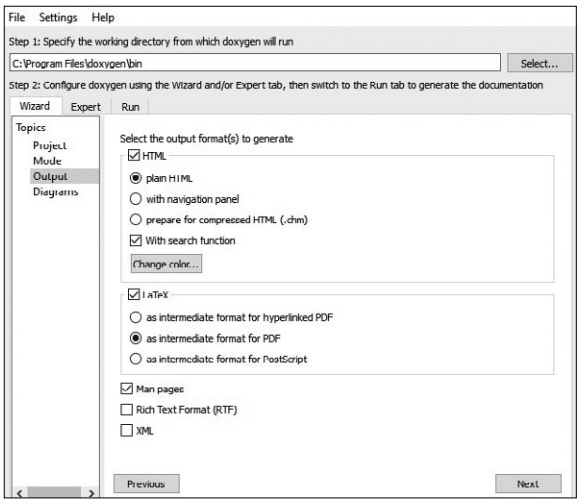


Figura 6.11
Output Doxygen

La última ventana sería la de *Diagrams*. Lo útil sería seleccionar *Use built-in class diagram generator*, y se pulsa la opción *Next*. Ahora llega el momento de ejecutar Doxygen para permitir generar la documentación en HTML, por lo que se pulsará el botón *Run Doxygen*. En esta ventana se tiene la opción de mostrar toda la configuración mediante parámetros y también almacenar el log de salida (figura 6.12).

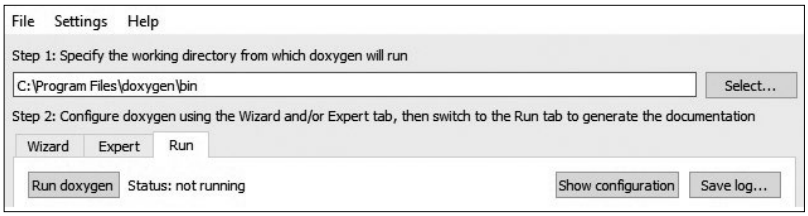


Figura 6.12
Run Doxygen

Por último, se obtendrá toda la documentación que se ha configurado en la aplicación y en Doxygen, y se visualizará una página web como la de la figura 6.13.

Figura 6.13
Documentación final

6.3. Formatos estándar para la documentación

En el mercado actual existe un conjunto amplio de formatos estándar para documentar, pero se establecen una serie de protocolos o medidas para que todo el mundo las lleve a cabo. De esta forma se pretende que cuando alguien lea el código entienda fácil e intuitivamente qué flujo lleva el código y cuál es su función a grandes rasgos.

Con relación a la nomenclatura, es crucial dar un nombre específico a las clases, métodos, variables, etc., para poder posteriormente identificar cualquier variable o clase dentro del código. Por ejemplo, a la hora de nombrar las variables sería interesante poner un identificador inicial que haga referencia al tipo de variable. Si se tiene un integer y la variable almacena una posición dentro de un array, lo ideal sería nombrarla *iposicion* o *iposition,* dependiendo de si se está codificando el código en inglés o en español.

Toda la documentación y nomenclatura se realiza para que el código pueda ser reutilizado en un futuro, sin necesidad de estudiar el código en profundidad. Por lo que sería necesario documentar de una forma correcta la aplicación y cada una de las clases que la componen.

Por ejemplo, en Java o en cualquier lenguaje de programación es necesario comentar las líneas de código. Los comentarios no cambian nada el código implementado, sino que simplemente ayudan al programador a comprender mejor las líneas de código del proyecto software. Más adelante, se entrará en profundidad en comentar cómo documentar las clases y los métodos implementados. Pero es importante mencionar que existen dos tipos de comentarios:

a) Comentario en línea: este tipo de comentario va precedido de dos barras slash "//". Por ejemplo, atributos sería un comentario en línea:

```
//Atributos
    /**
     * Nombre del autor
     */
    private String nombre;
```

b) Comentario en multilínea o en bloque: estos comentarios van precedidos de "/*" y luego se escribiría texto y se finalizaría con "*/". Todo el texto que va incluido entre estos símbolos se considera comentario. Por otro lado, si se quiere que lo interprete Javadoc para que se procese como documentación final de la aplicación web, es diferente. Habría que englobar los comentarios entre los símbolos "/**" y "*/". Un ejemplo para que se observe sería de la siguiente forma:

```
/**
 * Clase Autor
 * Contiene información sobre el autor de los libros
 * @author javier
 * @version 1.0
 *
 */
```

Se han explicado varias herramientas de generación automática de documentación, pero todas tienen que tener un objetivo común, que es documentar de forma clara y concisa las clases. En primer lugar, se explicará cómo se documenta una clase, por ejemplo, de Java la lista de atributos serían los siguientes:

- ✓ *Nombre de la clase:* sería teclear el nombre de la clase de forma identificativa.
- ✓ *Descripción:* una descripción corta de qué contiene la clase.
- ✓ *Autor:* quién es el autor o autores de la implementación de la misma.
- ✓ *Versión:* es fundamental para comprobar cuántas versiones se han realizado de la misma, en caso de sufrir modificaciones.

Un ejemplo se puede observar en el siguiente código:

```
/**
 * Clase Autor
 * Contiene información sobre el autor de los libros
 * @author javier
 * @version 1.0
 *
 */
public class Autor {
```

A continuación, en la implementación del código vienen los atributos de la clase, que se teclearán con un comentario inicial de la forma //Atributos, y después una descripción corta de cada uno de ellos. Como se observa en el siguiente código:

```
//Atributos
    /**
     * Nombre del autor
     */
    private String nombre;
     /**
     * Ciudad del autor
```

```
 */
private String ciudad;
 /**
 * Lista de libros
 */
            private List libros;
```

Si se sigue avanzando en la clase, es importante documentar el constructor o constructores, métodos y destructores de la clase. En este elemento sí habría que poner algunos campos más, y serían los siguientes:

- *Tipo de método:* público o privado.
- *Nombre del método o constructor:* descripción breve del método o constructor.
- *Parámetros:* las variables de entrada al método o constructor.
- *Variable de salida:* si el método devuelve algo.

Se puede observar en el siguiente código la documentación de constructor:

```
//Constructor
      /**
     *  Autor
     * @param nombre nombre del autor
     * @param ciudad ciudad del autor
     *
     */
    public Autor (String nombre, String ciudad)
    {
        this.nombre=nombre;
        this.ciudad=ciudad;
        this.libros=new ArrayList();
     }
```

Se puede observar en el siguiente código la documentación de un método público:

```
            //Métodos públicos
    /**
     *  setNombre
     * @param n nombre del autor
     */
    public void setNombre(String n){
            this.nombre=n;
    }
      /**
     *  getNombre
     * @return nombre nombre del autor
     */
    public String getNombre(){
        return this.nombre;
  }
```

Actividad propuesta 6.4

Toma una aplicación como ejemplo y documenta la clase y sus métodos, como se ha explicado anteriormente. De manera opcional, se puede generar la documentación mediante Javadoc.

6.4. Creación y utilización de plantillas

Se ha explicado cómo realizar documentación de forma manual. También se puede realizar una documentación estructurada, y que se observe de forma intuitiva y amigable, para lo que se usa Javadoc, phpDocumentor o cualquier otra herramienta de generación de documentación. Pero existe otra posibilidad que permite programar las clases en Java con una plantilla o template por defecto.

El objetivo de estas plantillas es ahorrar tiempo sin tener que escribir un código que es literal y repetitivo, además de comentarios, como, por ejemplo, el autor de la clase, la versión, etc.

En cualquier IDE, como Netbeans, se usan las plantillas de archivo, de proyecto, de clases, etc. También existe la posibilidad de las sugerencias de código. En nuestro caso, se van a explicar las plantillas de archivo que permitirán crear la clase con unos comentarios y código por defecto, muy útiles a la hora de programar una aplicación con una serie de clases.

A partir de un proyecto creado, se va a crear una clase nueva, y se observará cómo se crean elementos y códigos por defecto. Si se pulsa en un paquete del proyecto en el botón derecho y, a continuación, en Nuevo y por último en Java Class, se observa la ventana 6.14.

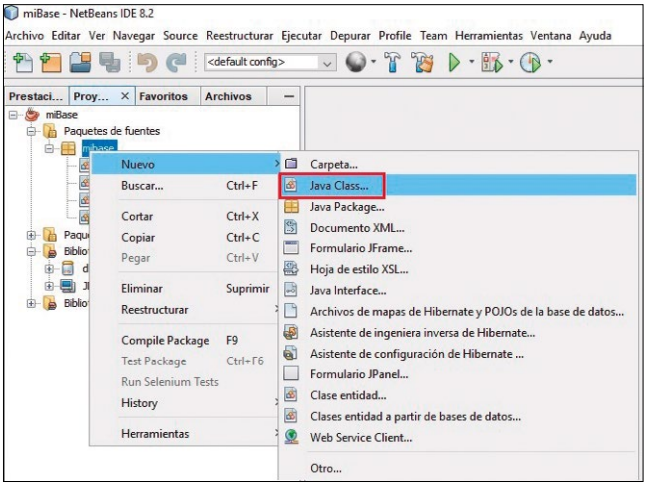


Figura 6.14
Java Class

Una vez seleccionada esa opción, será necesario darle un nombre a la clase (nuestro caso, Editorial). Después de terminar el asistente aparecerá un archivo creado a partir de una plantilla. El archivo creado es algo parecido al de la figura 6.15.

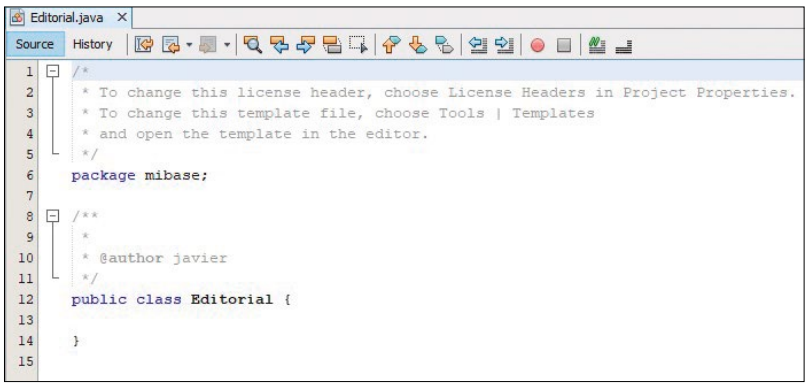


Figura 6.15
Clase Editorial

En la imagen anterior hay que distinguir dos partes, una de ellas es la cabecera o header, en la que se pueden cambiar las propiedades del proyecto. Y otra sería la plantilla que genera el archivo (nuestro caso). Para editar, modificar, duplicar la plantilla se tiene que seleccionar la opción Herramientas y dentro de esta la subopción Plantillas o Templates. Aparecerá una ventana como la 6.16.

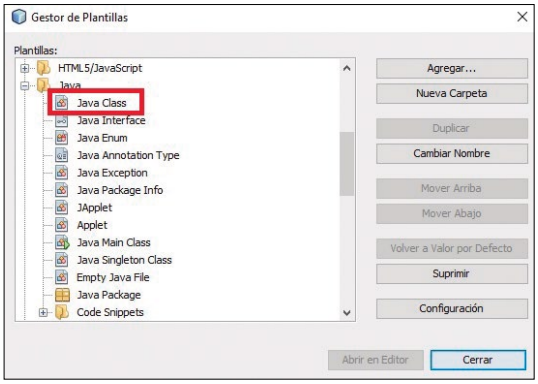


Figura 6.16
Plantillas

En esta ventana se pueden observar las diferentes plantillas que existen de los diferentes lenguajes de programación que soporta Netbeans 8.2. En nuestro caso, la explicación se focalizará en el lenguaje Java y, más concretamente, la creación de una nueva clase en Java (Java Class). Una plantilla se compone de la siguiente lista:

- *Nombre:* nombre de la plantilla.
- *Descripción:* breve descripción de la plantilla.
- *Cuerpo de la plantilla:* es un conjunto de texto fijo y variables que se pueden usar para que, cuando se cree un elemento del tipo de plantilla, se generen automáticamente. Algunas de las variables son las siguientes:

  - ${user}: inserta el nombre del usuario.
  - ${date}: inserta la fecha actual, con el formato 04-sep-2020.

- ${time}: inserta la hora actual en la cual se crea el fichero de la clase.
- ${encoding}: coloca la codificación de caracteres UTF-8.
- ${name}: visualiza el nombre del fichero sin extensión.

Para colocar las variables anteriores en la plantilla, se selecciona *Java Class* y luego *Abrir en Editor*. Con ello se visualizará una pantalla como la 6.17.

Si ahora se crea una nueva clase llamada Editorial, tomará como base la plantilla creada anteriormente, y el resultado será el de la figura 6.18.

Por otro lado, si se quieren modificar las propiedades del usuario, por ejemplo, su nombre, sería necesario pulsar en la opción Configuración de la figura 6.16, y se visualizará una pantalla como la 6.19.

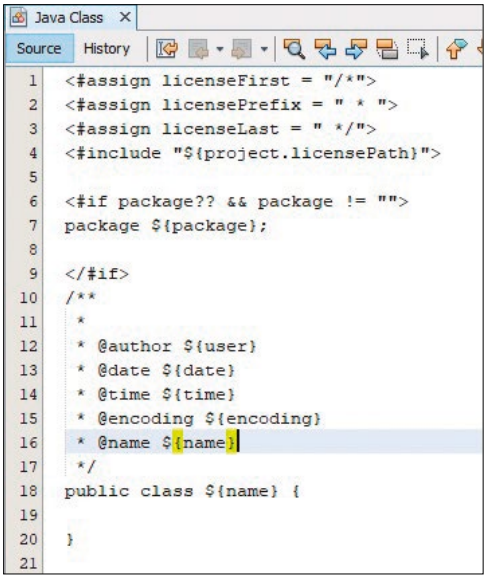


Figura 6.17
Plantilla Java Class

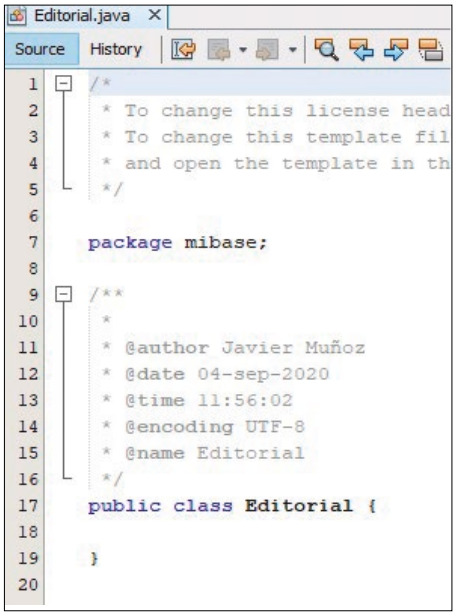


Figura 6.18
Clase Editorial

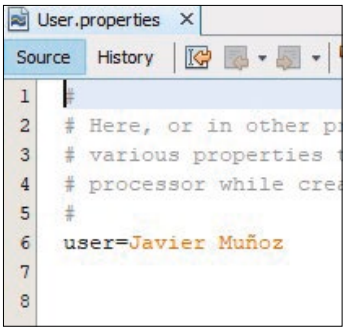


Figura 6.19
User.propierties

Recurso web

www

Accede a la Wiki de Netbeans para profundizar sobre las variables de las plantillas.

Además de las opciones comentadas, la ventana de las plantillas (figura 6.16) permite las siguientes opciones:

- ✓ *Agregar:* permite agregar un fichero donde esté definida una plantilla en la que el tipo de fichero debe ser .java.
- ✓ *Nueva carpeta:* crea una carpeta para generar plantillas.
- ✓ *Duplicar:* permite duplicar una plantilla ya creada.
- ✓ *Cambiar nombre:* cambia el nombre a una plantilla.
- ✓ *Mover arriba:* mueve hacia arriba una plantilla.
- ✓ *Mover abajo:* mueve hacia abajo una plantilla.
- ✓ *Volver a valor por defecto*: regresa a los valores por defecto antes de modificar nada.
- ✓ *Suprimir:* elimina una plantilla creada anteriormente.
- ✓ *Configuración:* permite editar las propiedades del usuario.

6.5. Herramientas colaborativas para la elaboración y mantenimiento de la documentación

Actualmente, en el mercado existe un abanico amplio para usar herramientas colaborativas para elaborar documentación, tanto open source como de pago. Hay varias herramientas que son bastante interesantes, pero creo que una de las mejores es Slack, que posee una versión bastante potente gratuita, aunque si se quiere ampliar habría que pagar alguna cantidad de dinero adicional.

Y, por otro lado, la mayoría de las empresas de ámbito nacional e internacional dedicadas a la implementación de software usan el paquete Microsoft Office 365, que posee una gran variedad de opciones, desde el paquete básico hasta el paquete Premium (esta no es gratuita).

6.5.1. Herramienta Slack

Con relación a Slack, es una de las herramientas que más se usa en el mercado actual, sobre todo en equipos de trabajo para compartir información y documentación. Posee una dificultad añadida, que es el amplio conjunto de posibilidades que tiene, pero en el momento en el que el equipo se acostumbra es realmente útil y práctico su funcionamiento.

Entre las ventajas que posee están las siguientes:

- Se puede compartir documentación, vídeos e imágenes desde tu equipo o desde Google Drive.
- Posee integración con otras aplicaciones del entorno de trabajo remoto como Trello, herramientas diarias como Gmail y Google Calendar, aplicaciones esenciales GotoMeeting y Salesforce, incluso Robots como Troops.
- Versiones de escritorio, web y móvil.
- Permite crear canales para controlar un proyecto determinado y ver los detalles de tal canal.
- Personalización de tus notificaciones, tiempo, sonido, etc.
- Contiene un generador de workflow, fundamental en la realización de un proyecto software o aplicación web.

Por todo ello, es una herramienta ideal colaborativa y lo suficiente potente para comenzar a trabajar. Si bien es cierto, que si necesitas más almacenamiento o mayores medidas de seguridad existen dos opciones de pago, que son las versiones Standard y Plus.

Para observar la herramienta se comienza con ingresar en la URL https://slack.com/signin y se visualizará una pantalla donde se tendría que ingresar tu email.

Una vez que se introduce el correo electrónico y se pulsa Continuar te has dado de alta en Slack. Posteriormente para evitar spam y ataques innecesarios te envían un código de seis números al correo electrónico que se ha tecleado. Al introducirlo en la URL que se visualiza aparecerá una página donde se tecleará el nombre del equipo software (figura 6.20).

En la siguiente página, te pedirá introducir alguien más del equipo de trabajo: simplemente se puede omitir o poner alguien que forme parte del equipo de trabajo, tecleando el email del mismo. También se puede enviar un enlace de invitación.

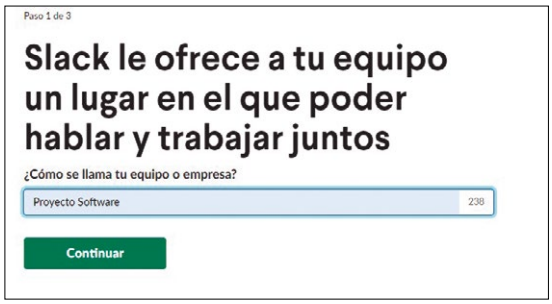


Figura 6.20
Equipo de trabajo

Figura 6.21
Mensaje al equipo

Por último, se envía un mensaje al equipo y se está listo para comenzar a trabajar (figura 6.23). En este momento, ya está el equipo preparado para comenzar el proyecto software. La página principal del proyecto es la que se muestra en la figura 6.22.

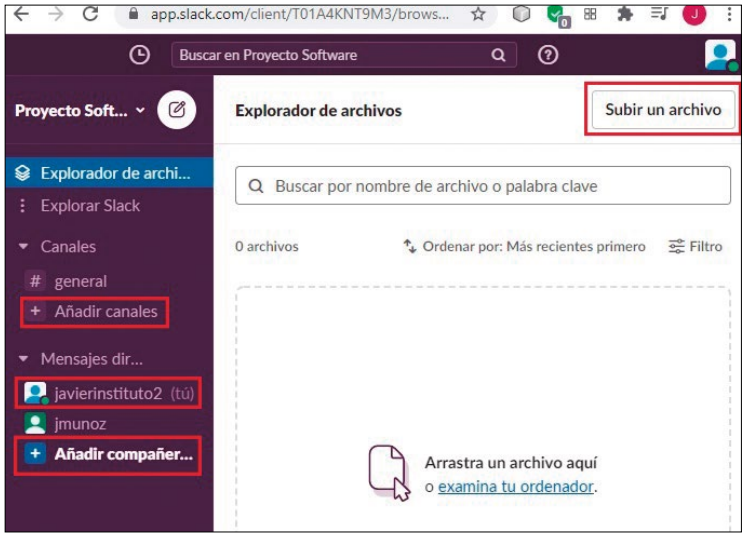


Figura 6.22
Proyecto Software

En la pantalla de la figura anterior se tiene una serie de opciones que se van a explicar a continuación:

- ✓ *Canales:* permite abrir un canal para un tema determinado o por un problema del proyecto o cualquier otra anomalía.
- ✓ *Equipo:* se puede añadir a compañeros de equipo en cualquier momento.
- ✓ *Archivos:* se pueden subir archivos de importancia para el proyecto.
- ✓ *Aplicaciones:* nos permite acceder a cualquier tipo de aplicación de las que se han visto anteriormente.

Este último apartado es interesante, ya que permite llegar a las aplicaciones a las que se puede acceder desde Slack. Si se pulsa en la opción *Explorar Slack* de la figura 6.22 se visualizará la pantalla 6.23.

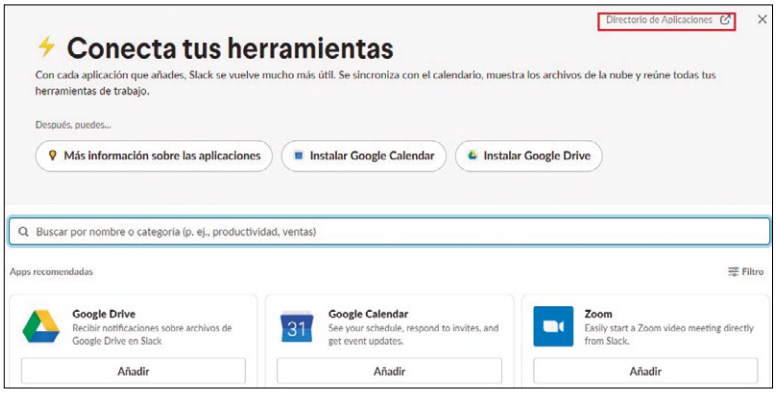


Figura 6.23
Aplicaciones Slack

Además, se puede navegar al directorio de aplicaciones y se puede observar el listado de aplicaciones disponibles.

Por otro lado, dentro del *Proyecto Software* de la figura 6.22 se puede observar la imagen 6.24.

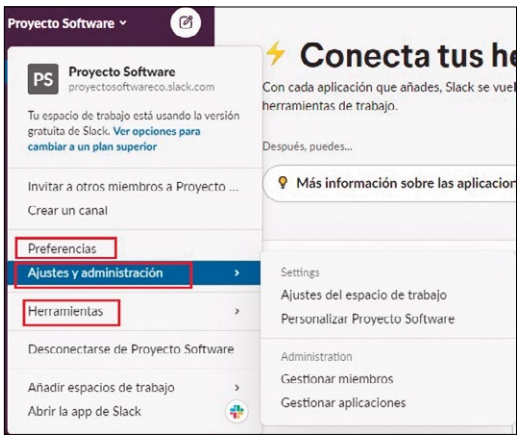


Figura 6.24
Configuración Slack

Dentro de esta lista de opciones, algunas de ellas ya se han comentado. Las principales son las siguientes:

a) Preferencias: permite configurar las notificaciones, la barra lateral de iconos rápidos, los temas que se pueden configurar, los mensajes y los medios, el idioma y la región, la accesibilidad, marcar como leído y avanzados que está relacionado con la opciones que existen para escribir código, para buscar alguna información en los canales, etc.
b) Ajustes y administración: está relacionado con los ajustes y autorizaciones del espacio de trabajo. Existen algunas opciones de esta ventana que son de pago como la autenticación o algunos permisos específicos, por ejemplo, las solicitudes en canales compartidos.
c) Herramientas: es una opción bastante interesante, dado que permite conocer los mensajes enviados, almacenamiento usado (límite 5 Gb) y aplicaciones e integraciones instaladas.

Actividad propuesta 6.5

Crea un espacio de trabajo en Slack que se llame Aplicación Mutante, y mediante dos correos electrónicos forma un equipo de trabajo para intercambiar mensajes, e instala alguna de las aplicaciones. Todo de forma gratuita para observar el manejo de esta herramienta tan útil.

6.5.2. Herramienta Microsoft Office 365

En el tejido productivo empresarial y sobre todo en el mundo de la informática, Office 365 es una de las herramientas punteras, pero tiene el inconveniente que es de pago totalmente, por lo que se sugiere usar la anterior, pero en caso de que exista presupuesto, se puede utilizar perfectamente.

Office 365 es un entorno de colaboración que aglutina una serie de aplicaciones cruciales a la hora de acceder, compartir documentos. Además de los ya conocidos, como Word, OneNote, Excel o Powerpoint. Algunas de las ventajas de este producto son las siguientes:

- *OneDrive:* es un lugar de almacenamiento seguro en la nube, donde se podrá acceder a la información en cualquier momento, siempre que se disponga de una conexión a Internet. Además permite realizar un backup, en caso de pérdida o deterioro de información.
- *Seguridad y compatibilidad:* la seguridad se basa en que la información está cifrada, por lo que impide la lectura de la información en cualquiera de las aplicaciones que la componen. En cuanto a la compatibilidad, se puede acceder desde cualquier navegador y sistema operativo.
- *Acceso:* se puede acceder desde cualquier dispositivo, desde un Smartphone, una Tablet, portátil, etc. Por lo que su uso se puede dar en cualquier momento.
- *Colaboración:* se puede trabajar con varias personas del equipo de trabajo al mismo tiempo sobre un documento de trabajo.

Para contratar esta herramienta sería tecleando la URL https://www.microsoft.com/es-es/microsoft-365?rtc=1, y partir de aquí sería contratar un plan y empezar a funcionar con esta gran herramienta.

Para concluir, se van a listar las aplicaciones que posee tal entorno de trabajo, además de las propias de Microsoft Office:

CUADRO 6.1
Otras aplicaciones

Aplicación	Descripción
Delve	Colaborar con otros usuarios para organizar la información.
Forms	Crear formularios y analizar los resultados obtenidos.
Kaizala	Tener una agenda de trabajo para el día a día, aplicación móvil.
Microsoft Teams	Colaborar en equipos de trabajo, crear canales, iniciar chats, llamadas y reuniones. Todo ello securizado.
OneDrive	Compartir, sincronizar y almacenar información con otros usuarios del equipo de trabajo.
OneNote	Realizar notas, dibujos y poder compartir tal información.
Planner	Obtener consejos de expertos para realizar un evento.
Publisher	Diseño de calendarios, boletines e información de marketing.
SharePoint	Punto de compartición de información sobre noticias, publicaciones, etc.
Skype Empresarial	Organizar reuniones online y llamadas para el equipo de trabajo.
Stream	Compartir vídeos y clasificarlos.
Sway	Crear todo tipo de documentos como textos, imágenes, etc.
To-Do	Agenda con eventos de vencimiento y sincronizada con Outlook.
Whiteboard	Lienzo digital para dibujar de forma libre.
Yammer	Conectarse con el equipo de trabajo y aprovechar al máximo su rendimiento.

6.6. Instalación, configuración y uso de sistemas de control de versiones en SO Windows

Antes de comenzar a instalar y configurar un sistema de control de versiones es necesario comentar conceptos básicos que permiten controlar las diferentes versiones del ciclo de vida de una aplicación web o un software implementado mediante cualquier lenguaje de programación.

Cuando no existían sistemas de control de versiones, se programaba en un directorio y, si se modificaba algo, era necesario realizar un backup del directorio. Esto se hacía para volver hacia atrás en caso de que la modificación no fuera la correcta. Lo que significa que la forma de trabajar en unos veinte años ha cambiado considerablemente y, en la actualidad, es más fácil programar y reutilizar un código anteriormente programado.

6.6.1. Conceptos básicos

El sistema de control de versiones permite la modificación de las distintas partes o componentes de una implementación de un software específico. La versión es el estado en el cual se encuentra el producto que se está desarrollando en un punto del ciclo de vida del proyecto software.

Es interesante conocer algunos conceptos vitales para comprender el funcionamiento del control de versiones y para que el equipo de trabajo conozca la nomenclatura de cómo se trabaja en control de versiones software. Para ello se van a definir los siguientes conceptos:

a) *Repositorio (repository):* el lugar de almacenamiento de las distintas versiones del software que está desarrollando el equipo de trabajo. Puede ser la parte de la interfaz, una base de datos, una conexión, todo lo relacionado con la implementación software.
b) *Rama (branch):* cada programador tiene su copia de trabajo en local y realiza modificaciones sobre una rama en concreto del proyecto software. Incluso si es necesario puedes descargarte una copia de todo el proyecto y modificar la rama del proyecto asignada sin perjudicar a nadie. En caso de que los cambios sean correctos, se pueden subir al repositorio notificando el cambio.
c) *Conflicto:* se produce cuando varios programadores han realizado varios cambios en la misma rama del software y son contradictorios. Por lo que el control de versiones te avisa del conflicto, y es necesario solucionarlo hablando con tus compañeros de equipo. En caso de duda, el jefe de proyecto tendrá la última palabra.
d) *Cambio (change):* se realiza cuando se ha modificado la parte asignada a un programador y es subida al repositorio controlando este cambio el control de versiones.
e) *Revisión (revision):* es una versión del software y permite al sistema control de versiones controlar las distintas revisiones del software realizadas. Normalmente, se identifica con un contador o también con alguna etiqueta identificativa.
f) *Confirmar (commit):* confirmación de que se han realizado varios cambios en el mismo documento.

6.6.2. Funcionamiento del control de versiones

El sistema de control de versiones funciona con un repositorio local o remoto, que es donde se encuentra todo el código del proyecto, y con un cliente que se conecta a tal repositorio. En esta conexión se producen los diferentes cambios, commit, y las distintas operaciones que ofrece el cliente sobre el repositorio. Algunas de las herramientas que existen en el mercado son Subversion, Mercury, Git, Perforce, etc.

Existen varias formas de trabajar, dependiendo de la localización del repositorio y de la infraestructura que posea la empresa o el equipo de trabajo que use este tipo de sistemas. Se pueden clasificar los sistemas de control de versiones atendiendo a la arquitectura usada en tres tipos:

A) Locales

Es la versión más simple de la arquitectura de los sistemas, ya que la copia del proyecto software no se comparte con nadie. Es como si se hiciera un backup en local. Pero es la base de los siguientes tipos. Es posible hacerse una idea de esta arquitectura al observar la figura 6.25:

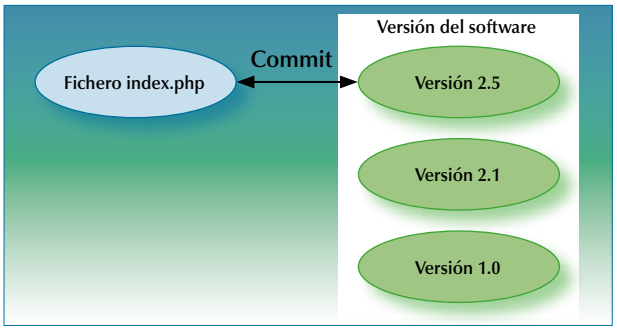


Figura 6.25
Sistema local

B) Centralizados

Esta arquitectura consiste en un servidor central que contiene el repositorio con todo el código, y este está asignado a un usuario, que normalmente es el jefe del proyecto software o la persona encargada de dar el visto bueno. Todas las operaciones que se hagan, como ramificaciones, o modificaciones de gran calado, pasan por su aprobación. Ejemplos de ellos son Github, Subversion, etc. Se puede ver la forma de trabajar observando la siguiente figura:

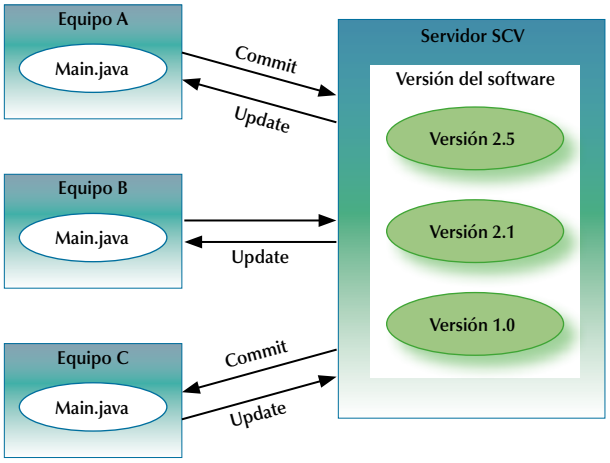


Figura 6.26
Sistema centralizado

C) Distribuidos

En este caso, la noción de repositorio central no existe, sino que cada equipo de trabajo tiene su propio repositorio. Los repositorios que existen en cada equipo se pueden intercambiar y fusionar revisiones entre ellos. Debería existir un repositorio disponible, a partir del cual se pueden sincronizar los demás. En esta opción es necesario que el equipo de trabajo controle perfectamente esta forma de trabajar, de lo contrario, pueden ocurrir interferencias. Tiene la ventaja de que no es necesario conectarse al servidor central. Por ejemplo, Git puede trabajar de esta forma. Para que el programador se haga una idea, la arquitectura sería la siguiente:

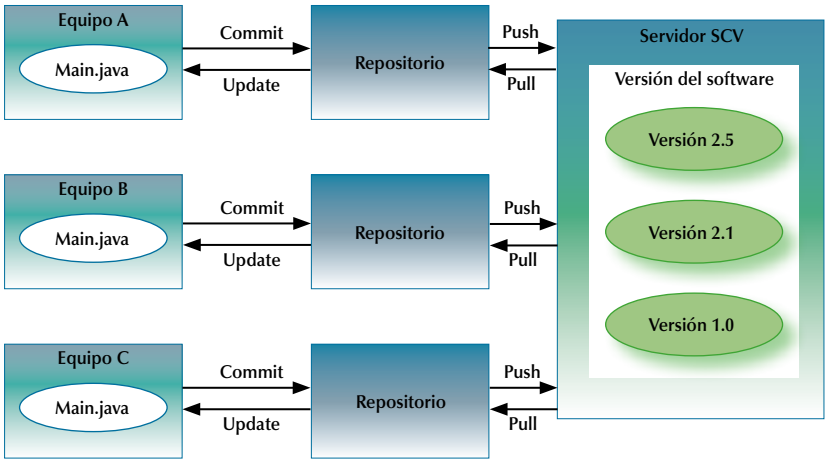


Figura 6.27
Sistema distribuido

PARA SABER MÁS

Se puede consultar este artículo de la página web de Git para profundizar sobre los sistemas de control de versiones centralizados.

6.6.3. Instalación y configuración de Git en Netbeans

Antes de comenzar con la instalación, se necesita recordar que se va a configurar una arquitectura del tipo distribuida, en la que el cliente va a ser Git en Netbeans y el repositorio central Github.com, que se puede acceder mediante la URL https://github.com/.

A partir de estas nociones, se va a configurar e instalar el sistema de control de versiones en Netbeans 8.2 y Github de la siguiente forma:

1. Lo primero sería dar de alta un usuario en Github mediante un login que no exista, un correo electrónico y una password. Posteriormente, Github te hace algunas preguntas y verifica que eres quien dices ser mediante el envío de un correo al email que se ha configurado en la página (figura 6.28).

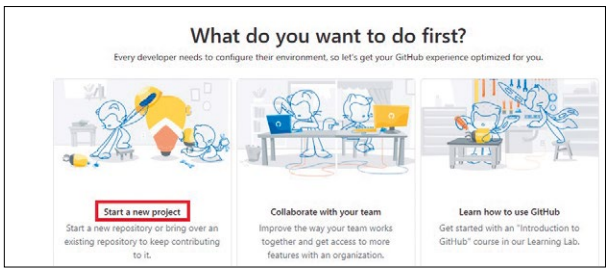


Figura 6.28
Github

2. Se pulsa *Start a new Project* y se visualizará una página como la de la figura 6.29.

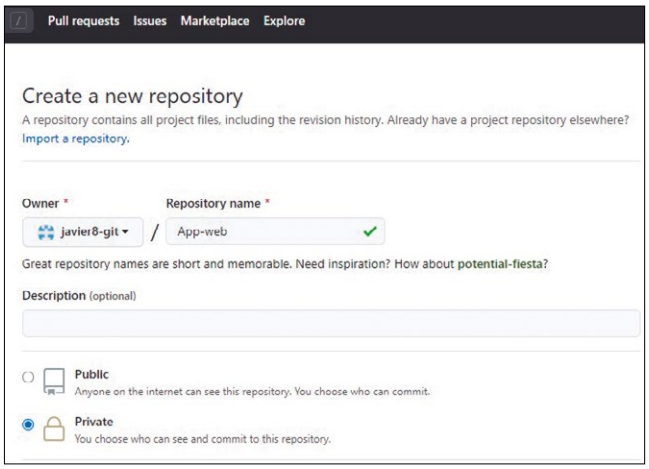


Figura 6.29
Creación de repositorio

3. En la figura 6.29 se visualizan varias opciones. La URL da la opción de importar un repositorio ya existente, el nombre de nuestro repositorio (App-web); muestra dos opciones, Private o Public. Si nuestro repositorio es público: cualquier persona de Internet puede ver nuestro repositorio, pero el propietario del repositorio acepta o no las modificaciones. Y, por otra parte, si es privado, el propietario elige quién observa el repositorio. En nuestro caso, se va a elegir Private. También Github da la opción de inicializar el repositorio con un Readme, de ignorar una lista de ficheros e incluso elegir una licencia. Con las opciones anteriores seleccionadas, se pulsa en Create repository, y se visualizará una pantalla como la 6.30.

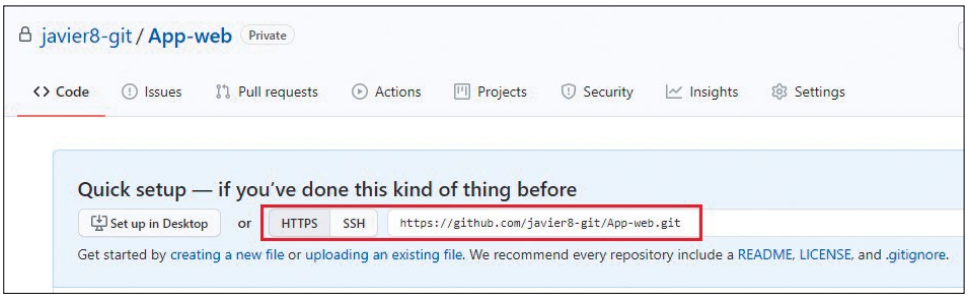


Figura 6.30
Repositorio App-web

4. Dos conceptos se van a tratar, el primero de ellos es que para sincronizar nuestro proyecto con el repositorio se configura en el escritorio o se copia la URL que está en el cuadro rojo y se configura posteriormente en Netbeans, como se explicará más adelante. Es necesario o recomendable crear un Readme, License y una lista de ficheros para ignorar en caso de que existiese. En nuestro caso, se copiará la URL para configurarla posteriormente en Netbeans.
5. El siguiente paso sería abrir un proyecto (en nuestro caso, Java) y, a continuación, sería abrir la opción *Team,* después la subopción Git y, por último, la elección Clone… Una vez seleccionada esta opción, se visualizará una pantalla como la 6.31.

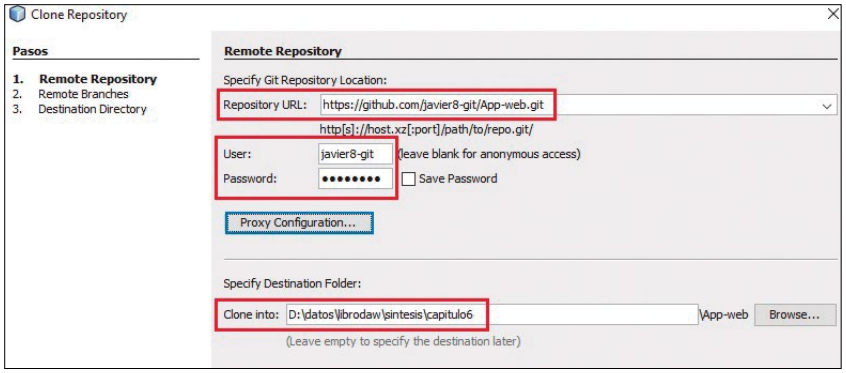


Figura 6.31
Configuración Netbeans

6. En la figura 6.32 se puede observar la configuración que se ha realizado. Lo primero de todo sería teclear la URL del repositorio que se ha copiado de Github. Por otro lado, hay que teclear el usuario y la password del repositorio. Se tiene la opción de configurar un proxy y de probar si la conexión funciona correctamente. Y, por último, se puede configurar un directorio de clonado. Ya se está en disposición de pulsar en Next, y se visualizará una pantalla para seleccionar las ramas del software. Como es la primera vez que se sincroniza, no es necesario seleccionar nada, por lo tanto, se pulsa de nuevo Next. Se visualizará la pantalla 6.32.

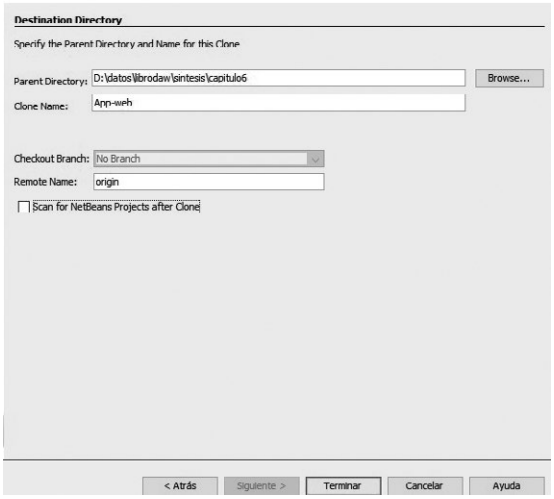


Figura 6.32
Configuración Netbeans II

Actividad propuesta 6.6

Configura Git con el repositorio Github o con cualquier otro y prueba que funciona correctamente.

7. Una vez configurado Git, se inicia el repositorio en la opción Initialize Repository, y se visualizará la pantalla 6.33.

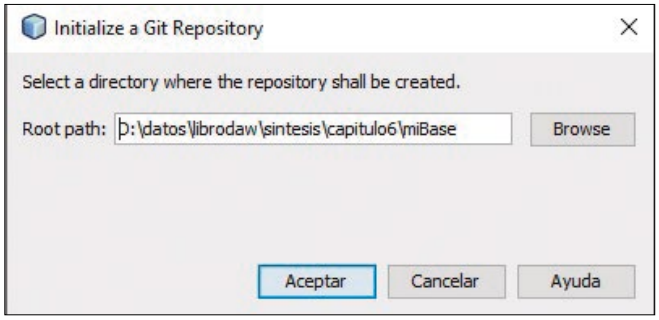


Figura 6.33
Inicializar repositorio

8. El siguiente paso es pulsar en el nombre del proyecto con el botón derecho del ratón y seleccionar Git y, posteriormente, Commit. Se visualizará la ventana 6.34.

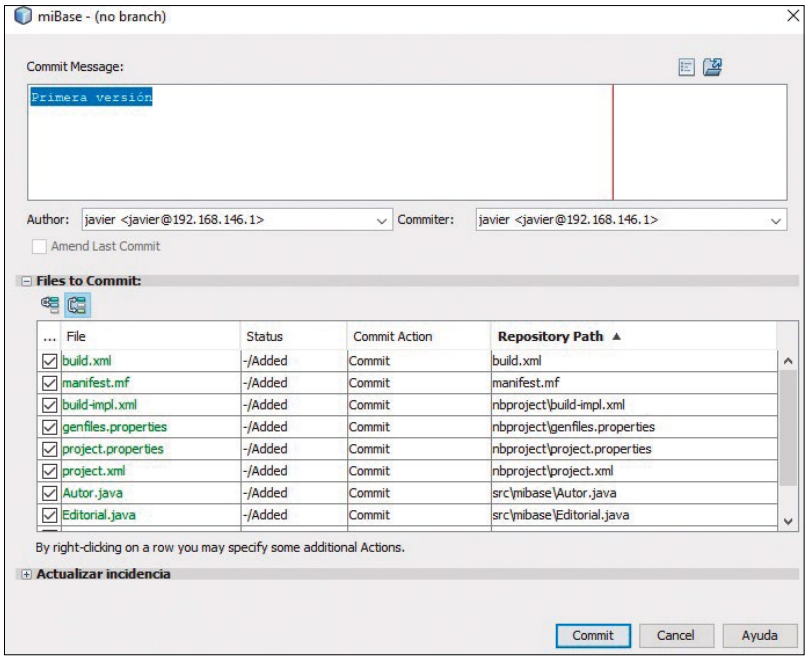


Figura 6.34
Commit

9. Será necesario realizar un push para subir el código al repositorio central. Esta operación se puede realizar de varias formas, una de ellas sería botón derecho del ratón pulsando en el proyecto y Git → Remote- → Push. Si se pulsa la opción anterior, se observará la ventana 6.35.
10. En la figura 6.35 se selecciona la rama master para poder enviarla al repositorio central de Github. Se observará otra pantalla parecida en la que se seleccionará también la opción de master. Si se pulsa Siguiente, se observará la pantalla 6.36.
11. En la figura anterior está preguntando si se creará una rama master o principal localmente, y si se quiere configurar esta rama para controlar la rama remota. Por lo que la respuesta es Sí. Con la operación anterior, ya está el código subido del proyecto a Github.Y se tiene la primera versión del software (figura 6.37).

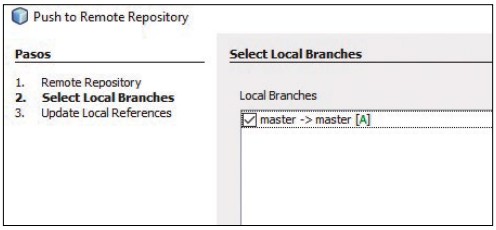


Figura 6.35
Operación Push

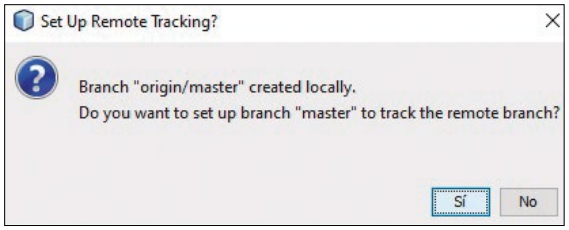


Figura 6.36
Operación Push

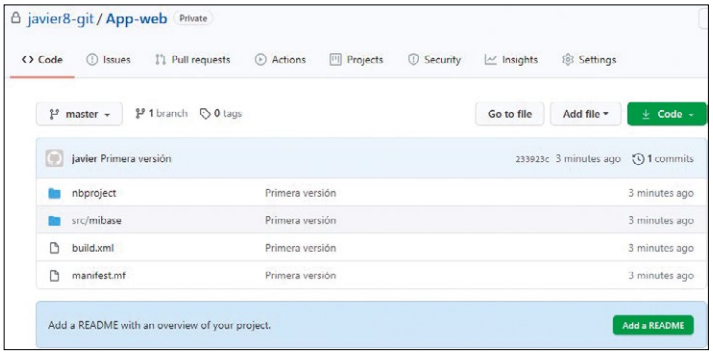


Figura 6.37
Github

Actividad propuesta 6.7

Una vez configurado Git en Netbeans, toma como ejemplo un proyecto y súbelo por primera vez a tu repositorio en Github u otro sistema de control de versiones.

6.7 Operaciones avanzadas

Git permite todo tipo de operaciones relacionadas con el código, por lo que se van a explicar las más importantes y avanzadas. Antes de comenzar, se van a explicar algunos de los colores que se asignan a ficheros según la acción realizada sobre ellos:

- Mibase.java: no han existido cambios.
- Mibase.java: el fichero ha sido modificado localmente.
- Mibase.java: se han añadido más líneas de código en el fichero.
- Mibase.java: existe un conflicto en las versiones del fichero.
- Mibase.java: el fichero es ignorado por Git y no será incluido en los comandos del sistema de control de versiones.

Las operaciones son las siguientes:

a) *Comparar versiones de software:* esta opción se realiza a partir de la opción *Team → DIFF*. Dentro de esta opción existen cuatro subopciones: *Diff to Head, Diff to Tracked, Diff to*

Repository Head y Diff to… La primera opción permite comparar la versión actual del software (situada a la derecha del panel) con la versión subida en el repositorio (figura 6.38).

En la figura se puede observar cómo se comparan las dos versiones del fichero Mibase.java. En la versión de la parte izquierda se pueden observar dos líneas rojas que significan que han sido eliminadas de una versión a otra. Y las líneas azules significan que han sido modificadas desde la última versión.

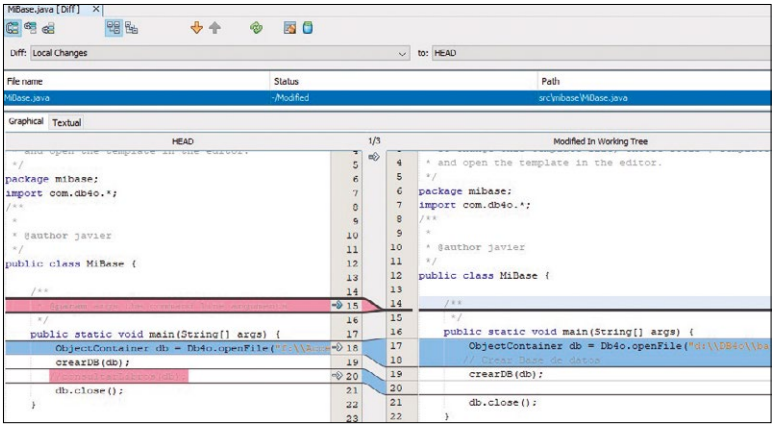


Figura 6.38
Diff

b) *Reverting Changes:* permite eliminar los cambios realizados. Sirve para volver hacia atrás en caso de algún error cometido en una nueva versión del software y, más concretamente de un fichero. La opción se selecciona mediante *Team → Revert Modifications.* Por ejemplo, si se borra un comentario y se graba para subirlo, y se necesita volver hacia atrás, esta opción lo permite.

c) *Branch (rama):* Git permite trabajar con ramificaciones del proyecto, incluso unirlas y borrarlas. Primero se va a crear una rama y a partir de ahí se puede realizar las operaciones que se han comentado. El objetivo de esta opción es el mantenimiento de varias ramas sobre un mismo código de base sobre el que se trabaja. Se parte del código que existe en la actualidad y se pulsa la opción *Team → Branch/Tag → Create Branch,* visualizándose la pantalla de la figura 6.39.

Se ha creado una nueva rama llamada 1-Etapa que se puede observar con el comando *Team → Checkout → Checkout Revision.* Dentro de la ventana que se observará es necesario seleccionar la opción Select, y se visualizará la imagen donde se observarán las distintas ramas y versiones que existen tanto en local como en re-

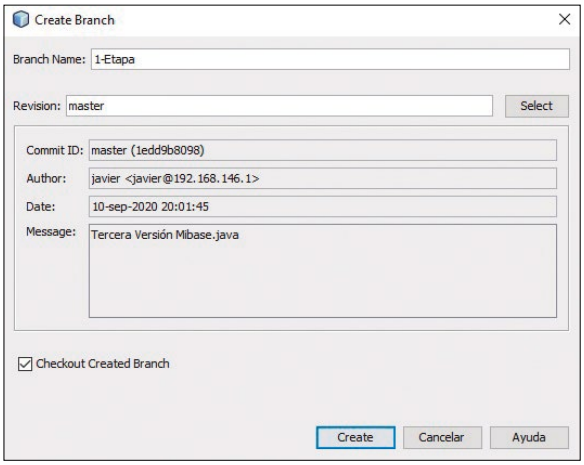


Figura 6.39
Branch

moto. Además, se visualizan los datos del autor del commit, la identificación del commit, la fecha y la descripción que se ha realizado (figura 6.40).

Si se observa atentamente en remoto no está todavía subida la rama creada denominada 1-Etapa. Siempre que se desee subir algo al servidor de control de versiones es fundamental seleccionar la opción de *Push*. En la figura 6.41 ya se pueden observar las dos ramas en Github:

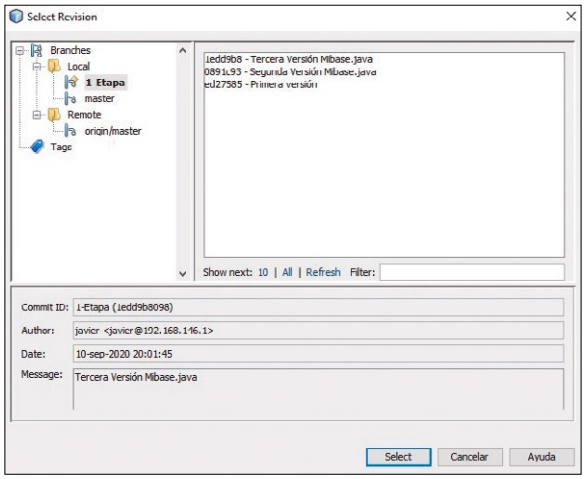


Figura 6.40
Checkout

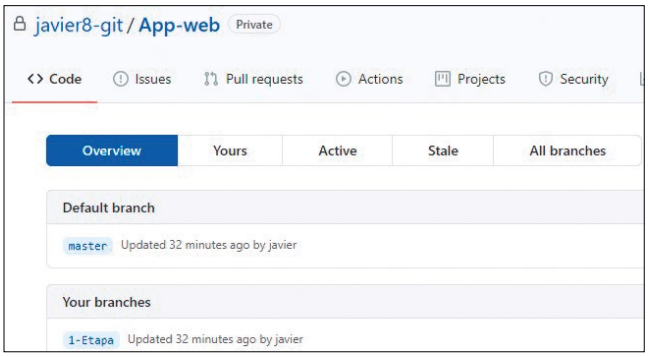


Figura 6.41
Branch

d) *Merge/patch:* se aplican los cambios realizados un fichero .java a otro. Normalmente, se utiliza cuando se tienen varias ramas, y algunas de ellas se necesitan tener en el mismo fichero.

e) *Conflict:* se produce en un equipo de trabajo cuando dos programadores modifican el mismo fichero o recurso y provocan que el cambio sea distinto en el mismo código. Github te avisa y se soluciona el problema aceptando el cambio más correcto para el desarrollo de la aplicación.

PARA SABER MÁS

Para profundizar con Git, sería necesario consultar este artículo de su página web.

6.8. Seguridad de los sistemas de control de versiones

Como se ha comentado anteriormente, Git se encuentra dentro de los sistemas de control de versiones distribuidos. En cada equipo de trabajo se tiene una copia del repositorio que se conecta a la red para acceder al repositorio Github central. Pero esta forma de trabajar puede ser cambiada en cualquier momento con la inclusión de, por ejemplo, Subversion. El acceso se realiza mediante dos protocolos que pueden ser configurados en Netbeans al inicio de la configuración:

6.8.1. Protocolo SSH

Permite la conexión entre dos equipos de forma segura. La implementación sería creando en Github un repositorio y seleccionando la opción SSH. Como se puede observar en la figura 6.42 y, más concretamente, en el recuadro rojo:

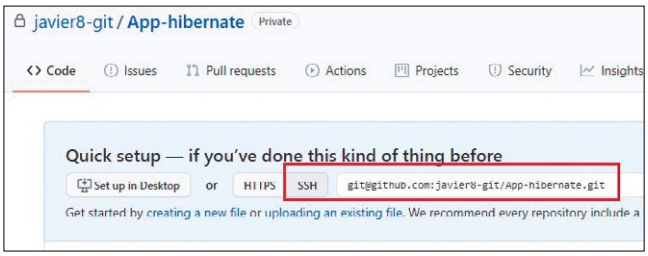


Figura 6.42
SSH

Es necesario configurar en Netbeans la opción de SSH seleccionada en el servidor de control de versiones. Para ello se configura la URL git@github.com:javier8-git/App-hibernate.git y el usuario javier8-git y la password que se ha creado en https://github.com/. La imagen será parecida a la de la figura 6.43.

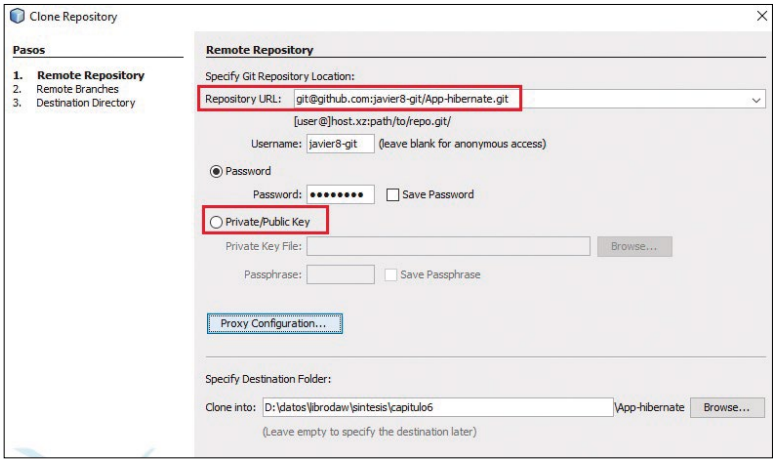


Figura 6.43
SSH Netbeans

Con esto terminaría la configuración de SSH y ya solamente se tendría que inicializar el repositorio, hacer el primer commit del proyecto correspondiente y realizar el comando push, y ya se copiaría el proyecto completo en el servidor de control de versiones. Además, existe la posibilidad de crear una clave privada para encriptar las comunicaciones (habría que crearla en local y también en el servidor). Por otro lado, también se pondría una palabra de paso que aumentaría la dificultad para descifrar la información.

6.8.2. Protocolo HTTPS

Esta opción permite realizar el mismo procedimiento que el anterior, pero en lugar del protocolo SSH sería con el protocolo HTTPS. Es similar al protocolo HTTP de páginas web, pero con la inclusión del protocolo SSL, que permite encriptar las comunicaciones entre el cliente y el servidor. Este procedimiento está explicado en las figuras 6.30 y 6.31.

Además de estas configuraciones, Git permite configurar los dos protocolos mediante un usuario y contraseña que permitirá solamente conectarse a aquellos programadores a los que se dé acceso. Así como la configuración de un proxy de salida a Internet por parte de la compañía o empresa en la que se esté desarrollando el software de la aplicación.

6.9. Historia de un repositorio

Después de haber llevado a cabo todas las operaciones anteriores, tanto básicas como avanzadas, Git permite la posibilidad de conocer todos los cambios realizados, ramas creadas y algunas opciones más. Para ello simplemente es necesario seleccionar el nombre del proyecto con el botón derecho del ratón y a partir de ahí *Git* → *Show History*. Aparecerán unas pantallas como las siguientes pulsando *Search* (figura 6.44):

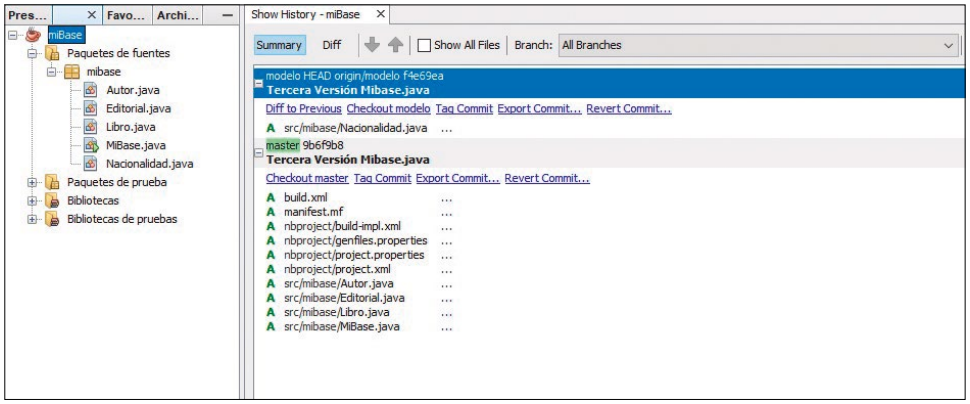


Figura 6.44
Show History

En la figura 6.44 se visualizan las dos ramas creadas, tanto master como modelo, en la que aparece que existe solamente una diferencia entre las dos ramas, es la creación del fichero Nacionalidad.Java. En la figura 6.45 se pueden observar todas las opciones para filtrar la información, como son las siguientes:

- *Message:* filtrado por mensajes escritos cuando se realiza un commit de una rama concreta.
- *Author:* es el programador del equipo de trabajo que ha realizado la operación commit.
- *Branch:* búsqueda por un nombre de una rama determinada. En nuestro caso, se tienen dos ramas, máster y modelo.
- *From and To:* entre fechas de operaciones realizadas sobre una rama determinada.

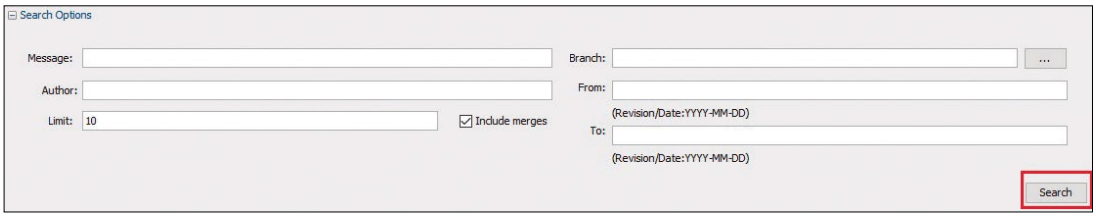


Figura 6.45
Búsqueda de eventos

Para concluir este apartado, se va a modificar el fichero Autor.java de la rama master para luego comprobar la diferencia con la otra rama llamada modelo mediante la visualización de Show History. En la ventana de Show History es necesario seleccionar DIFF y aparecerá una ventana como la siguiente, donde se puede observar la diferencia entre el fichero Autor.java de la rama master y la rama modelo (figura 6.46).

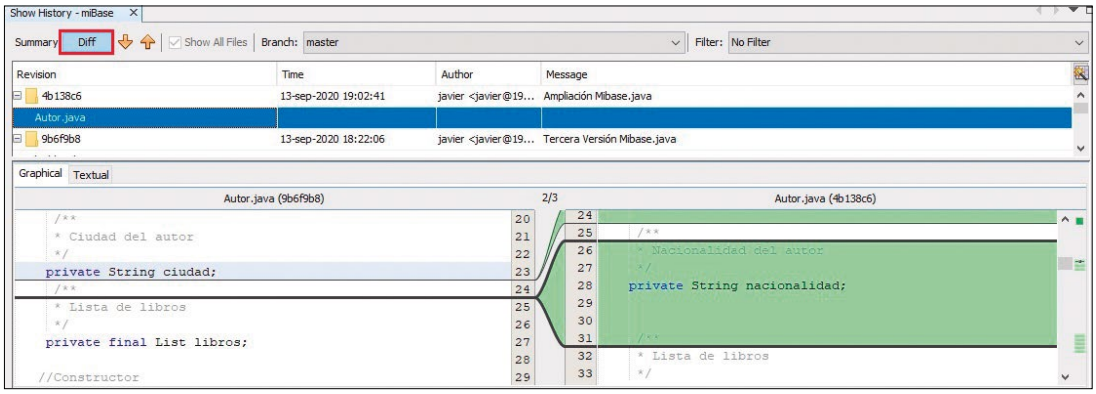


Figura 6.46
DIFF

Resumen

- En este capítulo final se ha explicado detalladamente la documentación relacionada con una aplicación web y el sistema de control de versiones para seguir el ciclo de vida a lo largo de su implementación a través de un equipo de trabajo.