

PLANT UML

ELMER CORTEZ – UO257192

JAIME LÓPEZ – UO257745

MARCOS ÁLVAREZ VIDAL – UO265180

SONIA GARCÍA LAVANDERA – UO263536

LUCÍA PRADO GARCÍA – UO265060



ÍNDICE

- Breve descripción de la funcionalidad del sistema.
- Principales atributos de calidad.
- Stakeholders.
- Restricciones.
- Issues.
- Módulos y componentes.
- Patrones y estilos arquitectónicos.
- Preguntas.

¿QUÉ ES PLANTUML?

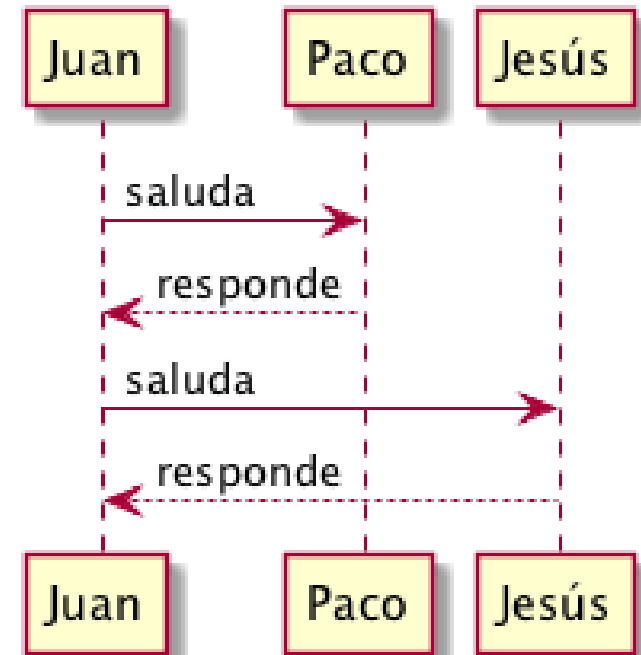
- Proyecto Open Source.
- Es una herramienta que nos permite escribir rápidamente diagramas.
 - Lenguaje simple e intuitivo.
 - Hecho en Java

EJEMPLO

@startuml

Juan -> Paco: saluda
Juan <-- Paco: responde
Juan -> Jesús: saluda
Juan <-- Jesús: responde

@enduml



EJECUTAR PLANTUML DESDE LÍNEA DE COMANDOS

- Primero, cree un archivo de texto con comandos PlantUML:

```
@startuml
```

```
Alice -> Bob: test
```

```
@enduml
```

- Segundo ejecutar PlantUML:

```
java -jar plantuml.jar diagramaDeSecuencia.txt
```

- diagramaDeSecuencia.png

MANERAS DE USAR PLANTUML



- Wikis y foros
- Editores de texto e IDEs
- Lenguajes de programación
 - Documentación
 - Editores en línea



ATRIBUTOS DE CALIDAD



MANTENIBILIDAD

- PROYECTO DE CÓDIGO ABIERTO (OPEN SOURCE)
 - BIEN ESTRUCTURADO



COMPATIBILIDAD



- Compatibilidad en todos los sistemas operativos.

SEGURIDAD

- El código malicioso se descubrirá rápidamente.





STAKEHOLDERS





¿QUÉ ES UN STAKEHOLDER?

"Stakeholder hace referencia a una persona, organización o empresa que tiene interés en una empresa u organización dada."

En otras palabras, un stakeholder en la gestión de un proyecto es todo aquel que es afectado por el proyecto, tanto de forma positiva como negativa sin importar si es afectado de forma directa o indirecta.

"Una buena planificación de proyectos debe involucrar la identificación y clasificación de los interesados, así como el estudio y la determinación de sus necesidades y expectativas."

Esto engloba tanto gente interna de la propia empresa (empleados, gerentes, propietarios) como de forma externa a dicha empresa o proyecto (proveedores, clientes, sociedad, otras empresas del sector) que se ven afectados por el desarrollo del proyecto.

TIPOS DE STAKEHOLDERS

- Accionistas
- Asociaciones (empresariales, industriales, profesionales, etc)
- Clientes
- Competidores
- Dueños
- Empleados
- Gobiernos
- Inversiones
- Medios de comunicación
- Proveedores

STAKEHOLDERS DE PLANTUML

- Desarrollador: Arnaud Roques.
- Equipo de desarrollo: Robert Brignull, Stefan Rotman, Konstantin Borisov, Steffen Dettmer, etc.
- Usuarios: Cualquier persona que utilice o integre PlantUML.
- Organizaciones: Gobiernos, entidades gubernamentales, etc.
- Lenguaje de programación: Java.
- Sistemas operativos: Linux, OS X, Windows.
- Aplicaciones que integran PlantUML: Atom, Doxygen, Eclipse, Google Docs, IntelliJ IDEA, LaTeX, Libre Office, Microsoft Word, NetBeans, Visual Studio Code, etc.
- Otros softwares de UML: MagicDraw, Papyrus UML, Modelio, ArgoUML, StarUML, etc.



ISSUES



GESTIÓN DE ISSUES

- ¿Issue?
- Agrupadas de 3 formas diferentes:

bug

feature-request

question



ISSUES MÁS RELEVANTES

- En PlantUML hay issues bastante relevantes como:
 - Arreglos visuales de algunos componentes en los diagramas.
 - Mejoras en la sintaxis del lenguaje.
 - Cómo añadir soporte para Jhipster.
 - Diferencias de renderizado de imágenes entre Linux y MacOS

ISSUES SOBRE LA DOCUMENTACIÓN

Multilingual Wiki Documentation



In an attempt to improve *PlantUML* documentation...

You are currently using `asciidoc` ▼ `syntax`

[Recent changes](#) [Wiki Toc](#) [View page history](#) [Reorder page](#) [Raw](#)

[Edit this part](#)

Issue about Namespace and Package

[Back to top](#)

Right now, the management of `namespace` and `package` may sound odd to users.

The real reason for this is some bad design decision made 10 years ago.

So here is our new proposal:

Merge the notion of package or namespace in PlantUML so that package or namespace will be synonymous and behave as in all regular programming languages.

- [Do not change anything](#) and keep the actual behaviour of PlantUML.
- [Ok, implement this](#) I understand that I *may* have to add `set separator none` to some old class diagrams to let them work as today.
- [Find another solution](#) that won't break ascending compatibility.

You just have to click on one of the buttons to vote.

You have to add `!pragma useNewPackage` directive to your diagrams to activate the new package behaviour.

Example: `classA --(InterfaceA) --> classB`

Anonymous 2020/02/04 04:01:26

I'm missing something. I vote "Ok, implement this". However, doesn't that mean that I understand I may have to add "set separator ." i.e. dot rather than none.

2020/02/09 11:49:45 **Me**

sdasdasd

Send



RESTRICCIONES



RESTRICCIONES: INSTALACIÓN / LENGUAJE / IMÁGENES

- A la hora de instalar localmente PlantUML debemos disponer:
 - Java
 - Graphviz
 - Puede ser opcional si solo necesitas diagramas de secuencia o de actividad
- Otro problema con el que nos podemos encontrar:
 - No funciona con todas las versiones de Graphviz
- Los diagramas se crean con el lenguaje específico de PlantUML
- Las imágenes se pueden generar en PNG, en SVG o en formato LaTeX
 - También es posible generar diagramas de arte ASCII (solo para diagramas de secuencia)

GRAPHVIZ

- Conjunto de herramientas de software para el diseño de diagramas definido en el lenguaje descriptivo DOT.
- Desarrollado por *AT&T Labs* y liberado como software libre con licencia tipo Eclipse.
- Programado en C
- Toma descripciones de gráficos en un lenguaje de texto simple y crea diagramas en formatos útiles, como imágenes y SVG para páginas web; PDF o PostScript para su inclusión en otros documentos; o mostrar en un navegador gráfico interactivo.
- Características útiles para diagramas concretos, como opciones de colores, fuentes, diseños de nodos tabulares, estilos de línea, hipervínculos y formas personalizadas.

LENGUAJE DE PLANTUML (DIAGRAMA DE CASOS DE USO)

@startuml

left to right direction

skinparam packageStyle rectangle

actor customer

actor clerk

rectangle checkout

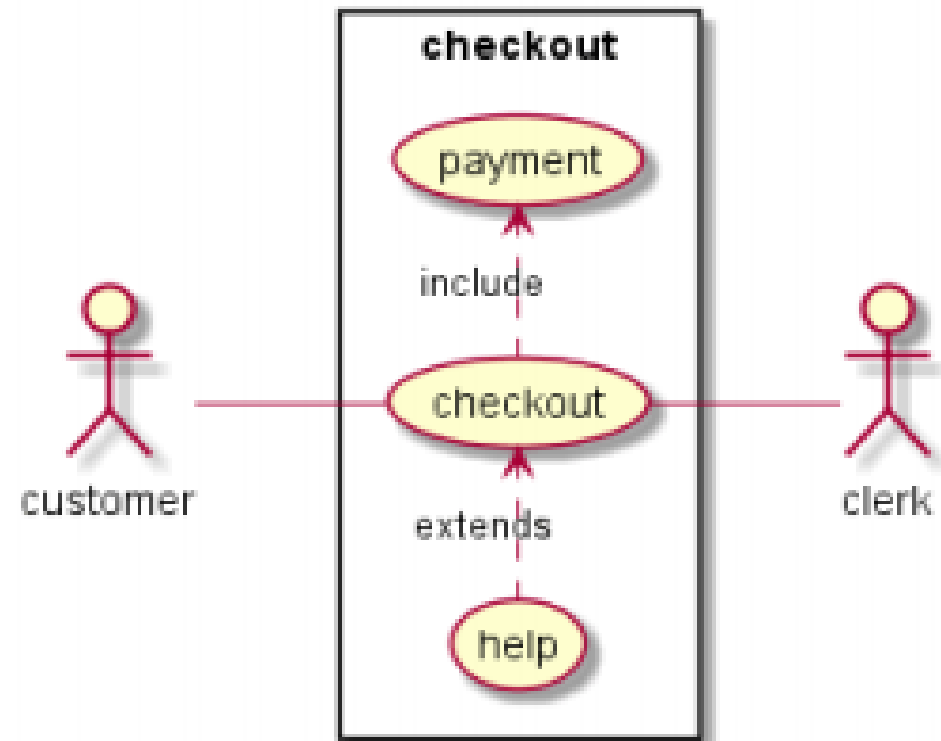
{ customer -- (checkout)

(checkout) .> (payment) : include

(help) .> (checkout) : extends

(checkout) -- clerk }

@enduml



LENGUAJE DE PLANTUML (DIAGRAMA DE SECUENCIA)

@startuml

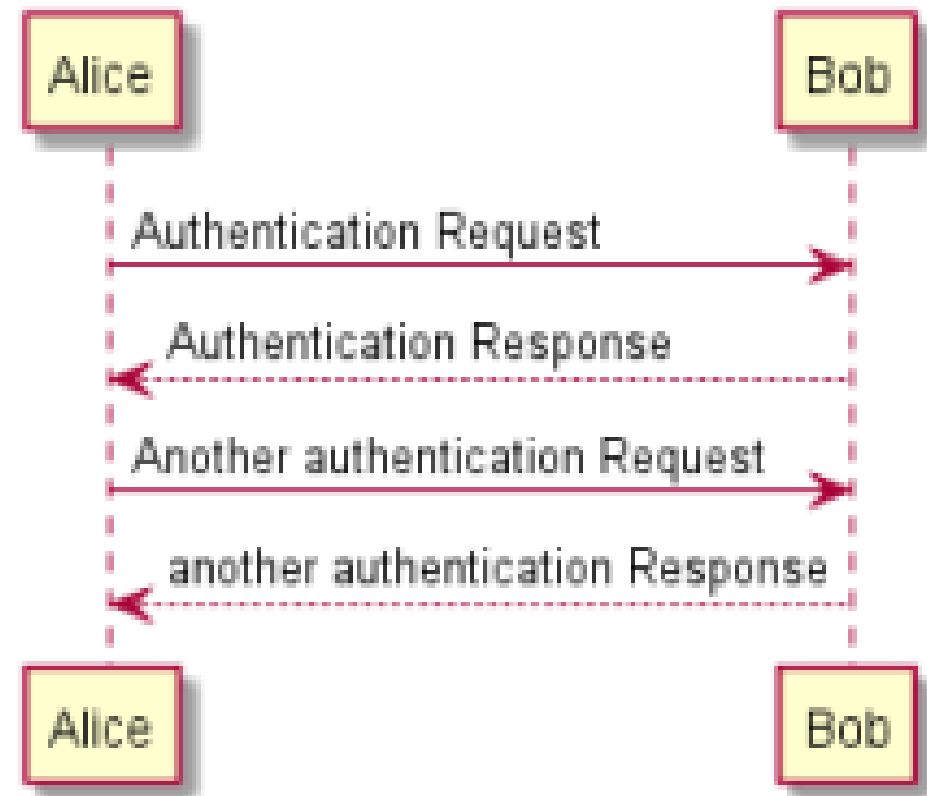
Alice -> Bob:Authentication Request

Bob --> Alice:Authentication Response

Alice -> Bob:Another authentication Request


Alice <-Bob: another authentication Response

@enduml





MÓDULOS Y COMPONENTES


































MÓDULOS Y COMPONENTES

- Estructura **modular**, donde se separan en paquetes las diferentes funcionalidades (componentes independientes) como ocurre con:
 - Servidor FTP y Telnet.
 - Exportación a SVG, PDF, PNG...
- Encontramos en la estructura de paquetes una sección donde se encuentran los comandos comunes además de la base para que cada módulo cree los necesarios (**command**).

- > pdf
- > png
- > posimo
- > postit
- > preproc
- > preproc2
- > project3
- > real
- > salt
- > sequencediagram
- > skin
- > sprite
- > statediagram
- > stats
- > style
- > sudoku
- > svek
- ▼ svg
 - > SvgData.java
 - > SvgGraphics.java
- > swing
- > syntax
- ▼ telnet
 - > AcceptTelnetClient.java
 - > TelnetServer.java

MÓDULOS Y COMPONENTES

- Cada tipo (o categoría) de diagrama está clasificada en su propio "módulo":
 - Sequence diagram (**sequencediagram**)
 - Class diagram (**classdiagram**)
 - Activity diagram (**activitydiagram** y **activitydiagram3**)
 - State diagram (**statediagram**)
 - Object diagram (**objectdiagram**)
 - Timing diagram (**timingdiagram**)
 - ASCII Math (**math**)
 - Y más...

- >  objectdiagram
- >  openiconic
- >  oregon
- >  pdf
- >  png
- >  posimo
- >  postit
- >  preproc
- >  preproc2
- >  project3
- >  real
- >  salt
- ▼  sequencediagram
 - >  command
 - >  graphic
 - >  puma
 - >  teoz
 - >  AbstractEvent.java
 - >  AbstractMessage.java
 - >  AutoNumber.java
 - >  Delay.java
 - >  Divider.java
 - >  DottedNumber.java
 - >  Englobber.java
 - >  Event.java
 - >  EventWithDeactivate.java
 - >  Grouping.java
 - >  GroupingLeaf.java
 - >  GroupingStart.java
 - >  GroupingType.java
 - >  HSpace.java



PATRONES USADOS



PATRONES USADOS

- En el proyecto encontramos varios patrones conocidos:
 - **Creacionales** como *Factory Method*, muy usado sobretodo para la creación de nuevos diagramas, o creación de diagramas a partir de comandos.
 - De **comportamiento**, tales como *Command*, que juega un papel muy importante y está presente en la mayoría de los módulos, aparece en lugares puntuales *Strategy* también.
 - Incluso podríamos decir que se usa el patrón **estructural** *Module* utilizado para implementar el concepto de módulos de software definidos por el paradigma de programación modular, en un lenguaje de programación que no lo soporta, o lo hace parcialmente.

PATRONES CREACIONALES

```
public abstract class UmlDiagramFactory extends PSystemAbstractFactory
public abstract AbstractPSystem createEmptyDiagram();

public class TimingDiagramFactory extends UmlDiagramFactory

    @Override
    public TimingDiagram createEmptyDiagram() {
        return new TimingDiagram();
    }
```

```
public abstract class PSystemBasicFactory<P> extends AbstractPSystem> extends PSystemAbstractFactory

    public P init(String startLine) {
        return null;
    }

    @Override
    public PSystemDefinition init(String startLine) {
        if (getDiagramType() == DiagramType.DEFINITION) {
            return new PSystemDefinition(startLine);
        }
        return null;
    }
```

TEMPLATE METHOD

```
public abstract class AbstractPSystem implements Diagram
final public ImageData exportDiagram(OutputStream os, int index, FileFormatOption fileFormatOption)
    throws IOException {
    final long now = System.currentTimeMillis();
    try {
        return exportDiagramNow(os, index, fileFormatOption, seed());
    }
    ...
}

protected abstract ImageData exportDiagramNow(OutputStream os, int index, FileFormatOption fileFormatOption,
    long seed) throws IOException;

@Override
protected ImageData exportDiagramNow(OutputStream os, int index, FileFormatOption fileFormatOption,
    long seed) throws IOException {
    final double margin = 10;

    final Scale scale = getScale();

    final double dpiFactor = scale == null ? 1 : scale.getScale(100, 100);
    final ImageBuilder imageBuilder = new ImageBuilder(new ColorMapperIdentity()
        null, false);
    final SkinParam skinParam = SkinParam.create(UmlDiagramType.TIMING);

    TextBlock result = getTexBlock();
}
```

```
public abstract class UmlDiagramFactory extends PSystemAbstractFactory
    protected abstract List<Command> createCommands();

private Step getCandidate(final IteratorCounter2 it) {
    final BlocLines single = BlocLines.single2(it.peek());
    if (cmds == null) {
        cmds = createCommands();
    }
    for (Command cmd : cmds) {
        ...
    }
}

@Override
protected List<Command> createCommands() {
    final List<Command> cmds = new ArrayList<Command>();

    addCommonCommands1(cmds);
    cmds.add(new CommandFootboxIgnored());
    cmds.add(new CommandRobustConcise());
    cmds.add(new CommandClock());
    ...
}
```

COMMAND

```
public interface Command<D extends Diagram> {

    CommandExecutionResult execute(D diagram, BlocLines lines);

public class CommandMultilinesFooter extends CommandMultilines<TitledDiagram>

@Override
public CommandExecutionResult execute(final TitledDiagram diagram, BlocLines lines) {
    lines = lines.trim(false);
    final Matcher2 m = getStartingPattern().matcher(lines.getFirst499().getTrimmed().getString());
    if (m.find() == false) {
        throw new IllegalStateException();
    }
    final String align = m.group(1);
    lines = lines.subExtract(1, 1);
    final Display strings = lines.toDisplay();
    if (strings.size() > 0) {
        HorizontalAlignment ha = HorizontalAlignment.fromString(align, HorizontalAlignment.CENTER);
        if (SkinParam.USE_STYLES() && align == null) {
```



¿PREGUNTAS ?

