



# ARQUITECTURA DEL SOFTWARE

2025-26

Jose Emilio Labra Gayo

Pablo González

Diego Martín

Celia Melendi



Escuela de  
Ingeniería  
Informática



Universidad de Oviedo

## Laboratorio 1

Uso de la IA en proyectos de  
desarrollo de software

# ● Como afecta la IA en el desarrollo

La forma de programar está cambiando

Cada vez surgen herramientas más potentes para completar código, explicar código, generar documentación, refactorizar, etc.

¿Cómo cambia el rol del arquitecto/desarrollador en un proyecto?

- Pasamos de escribir código línea a línea a diseñar y supervisar el código generado.
- Es un cambio de rol a más alto nivel
  - Problema: presupone un conocimiento técnico alto para asegurar la calidad del código generado.

# ● ¿Que puede hacer un asistente de IA?

## Algunas de las funciones que la IA puede realizar:

- Completar código
- Explicar partes de código ya escrito
- Generar documentación o tests
- Solución y explicación de errores
- Realizar tareas complejas en modo agente
  - Crear varios archivos de código
  - Crear archivos de configuración
  - Ejecutar rutinas (como por ejemplo, instalación de dependencias necesarias)

# ● Nivel 1: Autocompletado inteligente

## Con el autocompletado inteligente podremos:

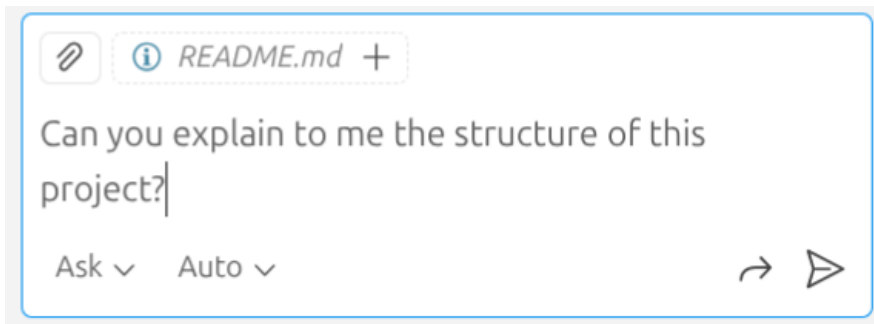
- Evitar escribir código repetitivo
- Muy útil para escribir trozos de código que pueden ser explicados fácilmente con un comentario
- En herramientas como VSCode, escribiremos el comentario y la IA empezará a generar código que podremos aceptar pulsando TAB

```
// Create a new method for authenticating users
app.post('/authenticate', async (req, res) => {
  const { username, password } = req.body;
  try {
    // Simulate a 1 second delay to mimic processing/network latency
    await new Promise((resolve) => setTimeout(resolve, 1000));
```

# ● Nivel 2: Explicación de código

## Con la explicación de código podremos:

- Hacer preguntas técnicas sobre un proyecto al completo.
- Entender partes de código no escritas por nosotros.
- En general, realizar preguntas que no impliquen una modificación directa del código de nuestro proyecto.



### Components

#### Webapp ( webapp )

A React single-page application built with Vite and TypeScript:

- **Main purpose:** User registration interface
- **Key files:**
  - `src/App.tsx` - Main app component
  - `src/RegisterForm.tsx` - Registration form
  - `vite.config.ts` - Build configuration
- **Testing:** Vitest unit tests + Playwright E2E tests with Cucumber
- **Deployment:** Dockerized Nginx server

#### Users Service ( users )



A REST API backend built with Node.js and Express:

- **Main purpose:** Handles user creation and returns welcome messages
- **Key endpoint:** `POST /createuser` - Creates a user and returns a greeting
- **Key files:**
  - `users-service.js` - Main API server
- **Monitoring:** Prometheus metrics + Grafana dashboard integration
- **Testing:** Vitest with Supertest for API testing



# ● Nivel 3: Modificación de código

## Con la modificación de código podremos:

- Realizar modificaciones complejas en múltiples archivos.
- Tenemos la posibilidad de aceptar o descartar los cambios propuestos
- Refinar el código generado iteraccionando con el sistema.
- Acceder a la cadena de razonamiento para entender como se ha generado el código.

  README.md +

add a new call to the users service to authenticate a user

Edit ▾ Auto ▾  


```
users > JS users-service.js > app.post('/authenticate') callback
40 app.post('/createuser', async (req, res) => {
41 });
42 app.post('/authenticate', (req, res) => {
43   const { username, password } = req.body;
44
45   if (!username || !password) {
46     return res.status(400).json({ error: 'Username and password'
47   }
48
49   // Simulate authentication (replace with real logic)
50   if (password.length >= 6) {
51     res.json({
52       success: true,
53       message: `User ${username} authenticated successfully`,
54       token: `token_${username}_${Date.now()}`
55     });
56   } else {
57     res.status(401).json({ success: false, error: 'Invalid crede
58   }
59 });
60
61
62 if (require.main === module) {
63   app.listen(port, () => {
64     console.log(`User Service listening at http://localhost:${po
65   })

```


# ● Nivel 4: Modo agente



## El modo agente:

- Realiza las mismas funciones que en el modo anterior pero además puede invocar **comandos de la terminal y otras herramientas**.
- Útil para tareas complejas que implican varios pasos.
- Los agentes pueden tardar varios minutos en realizar las tareas asignadas.

 README.md +

Can you configure kubernetes to deploy this project?

Agent ▾ Auto ▾ 

```
Run bash command?  
  
chmod +x /home/pablo/  
Proyectos/asw2526_0/k8s/*.sh  
  
Allow | ▾ Skip
```

Perfect! Now let me create one final comprehensive README for the k8s directory:

✓ Created README.md

Perfect! Let me verify the structure and create a summary:

✓ Read k8s

Excellent! Let me create one final file with quick reference commands:

✓ Created COMMANDS.md

Perfect! Now let me create a summary document:

✓ Created KUBERNETES.md

Perfect! Let me verify all files were created properly:

```
find k8s -type f -name  
"*.yaml" -o -name "*.md"  
-o -name "*.sh" | sort
```

k8s		
\$	cleanup.sh	🔍
↓	COMMANDS.md	🔍
\$	deploy.sh	🔍
↓	DEPLOYMENT.md	🔍
!	gamey-deploym...	🔍
!	gamey-service.y...	🔍
!	grafana-deploy...	🔍
!	grafana-service....	🔍
!	ingress.yaml	🔍
!	kind-config.yaml	🔍
!	kustomization.y...	🔍
↓	LOCAL_DEVELO...	🔍
!	namespace.yaml	🔍
\$	port-forward.sh	🔍
!	prometheus-co...	🔍
!	prometheus-de...	🔍
!	prometheus-ser...	🔍
🔍	README.md	🔍
!	users-deployme...	🔍
!	users-service.yaml	🔍
!	webapp-deploy...	🔍
!	webapp-service....	🔍



# ● Riesgos y buenas prácticas

## Calidad del código generado

- No tenemos ninguna garantía de que el código generado sea correcto.
- Debemos ser capaces de entender y supervisar el código generado para garantizar la calidad de software generado.

## Dependencia

- Utilizar generadores de código limita nuestra capacidad actual de escribir código.
- Este efecto se ve amplificado en la fase de aprendizaje/formación de un programador.

## Reglas básicas

- Si no lo entiendes, no lo uses.
- Úsalo para aprender preguntando, recibir explicaciones y mejorar como programador.
- Utilízalo para tareas repetitivas que ya sabes realizar.
- Añadir pruebas (código sin pruebas = código peligroso)



# Enlaces

- Code review with AI
  - <https://addyosmani.com/blog/code-review-ai/>
- Your job is to deliver code that Works:
  - <https://simonwillison.net/2025/Dec/18/code-proven-to-work/>