

🧩 Escenario

Tu equipo de desarrollo necesita migrar su proceso de trabajo a Azure DevOps. Como líder técnico, debes:

1. Configurar un proyecto completo en Azure DevOps con todos sus componentes.
2. Establecer un flujo de trabajo ágil con sprints y seguimiento de tareas.
4. Configurar políticas de calidad de código.
5. Establecer un proceso de code review mediante Pull Requests.
6. Documentar las decisiones tomadas y justificarlas.

📋 Tareas que debes cumplir

1. Configuración inicial del proyecto

- Crear una organización en Azure DevOps



- Crear un proyecto con la metodología ágil que consideres apropiada
- Configurar los equipos y áreas del proyecto
- Documentar en `decisiones.md` por qué elegiste esa metodología

Decisiones del Proyecto

Metodología

Usamos **Agile** porque es simple, flexible y nos deja adaptarnos rápido a cambios. Hicimos sprints de 2 semanas para entregar avances chicos y continuos.

Áreas y Equipos

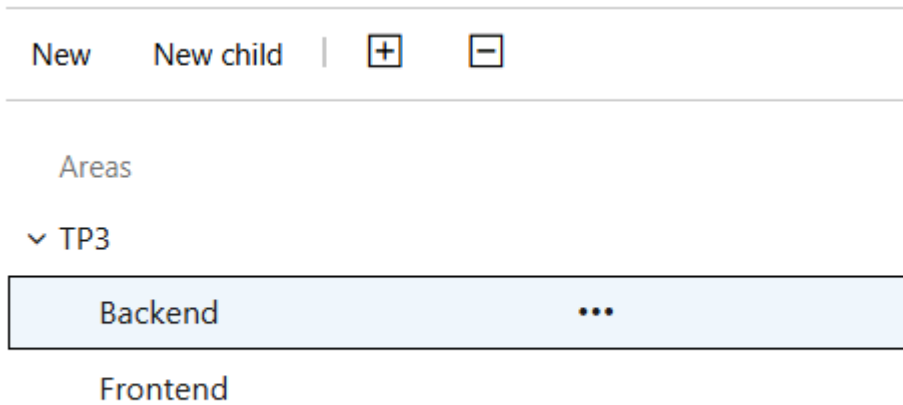
Dividimos el proyecto en dos sub-áreas: **Frontend** y **Backend**. Creamos un equipo para cada una. Así cada grupo ve solo lo suyo y es más fácil medir el progreso.

Estructura de Trabajo

- **Epic**: Sistema de Autenticación de Usuarios
 - **User Stories**: Registro, Login y Recuperar contraseña
 - **Tasks**: 2 por cada historia (formularios, endpoints, validaciones, etc.)
 - **Bugs**: Ejemplos de errores en validación y tokens

Justificación

La idea fue organizar el trabajo en pasos claros: Epic > Stories > Tasks. Esto nos da orden, visibilidad y permite entregar valor rápido.



2. Gestión del trabajo con Azure Boards

- Crear al menos:
 - 1 Epic que represente una funcionalidad completa
 - 3 User Stories relacionadas con el Epic
 - 2 Tasks por cada User Story
 - 2 Bugs de ejemplo
- Configurar un Sprint de 2 semanas

Iteration	Start Date	End Date
TP3\Iteration 1	22/9/2025	5/10/2025

- Asignar los work items al Sprint

ID	Title	Assigned To	State	Area Path
11	Registro de Usuario (Backend)	Unassigned	New	TP3\Backend
12	inicio de sesion	Unassigned	New	TP3\Frontend
10	Registro de Usuario (Frontend)	Unassigned	New	TP3\Frontend
9	Autenticacion de Usuarios	Unassigned	New	TP3\Backend
14	Diseñar formulario de registro	Unassigned	New	TP3\Frontend
13	Crear Endpoint POST	Unassigned	New	TP3\Backend
16	Token de sesion expira antes de lo esperado	Unassigned	New	TP3\Frontend
15	Validacion de email incorrecto en formulario de registro	Unassigned	New	TP3\Frontend

se lo sigamos a todos

- Documentar la estructura de trabajo elegida

Estructura de trabajo en Azure Boards

Para organizar el proyecto usamos la siguiente jerarquía:

- **Epic**

- Autenticación de Usuarios (funcionalidad completa)

- **User Stories**

- Registro de Usuario (Frontend)

- Registro de Usuario (Backend)
- Inicio de Sesión (Frontend)
- Inicio de Sesión (Backend)

- **Tasks** (2 por cada User Story, ejemplos:)
- Frontend: Diseñar formulario, validar campos
- Backend: Crear endpoint, validar en base de datos

- **Bugs**
- Validación de email incorrecta en formulario de registro
- Token de sesión expira antes de lo esperado

- **Sprint**
- Sprint 1 (22/09/2025 – 05/10/2025) → todas las User Stories, Tasks y Bugs fueron asignados aquí.

Justificación

Elegimos esta estructura porque nos permite:

- Dividir el trabajo en pasos claros (Epic → User Stories → Tasks).
- Separar responsabilidades entre **Frontend** y **Backend**.
- Identificar errores reales con **Bugs** de ejemplo.
- Planificar entregas en un **Sprint de 2 semanas** que da visibilidad y control sobre el avance.

3. Control de versiones con Azure Repos

- Importar o crear un repositorio con código de una aplicación
- Configurar políticas de branch para la rama principal:
 - Requerir Pull Request
 - Mínimo 1 reviewer



On

Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

1

- ☒ Allow requestors to approve their own changes
- ☐ Prohibit the most recent pusher from approving their own changes
- ☐ Allow completion even if some reviewers vote to wait or reject
- ☐ When new changes are pushed:

On

Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

Required

Block pull requests from being completed unless they have at least one linked work item.

Optional

Warn if there are no linked work items, but allow pull requests to be completed.

Off

Check for comment resolution

Check to see that all comments have been resolved on pull requests.

On

Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

Allowed merge types:

Basic merge (no fast-forward)

Preserves history exactly as it happened during development

Rebase and fast-forward

Creates a linear history by replaying the source branch commits onto the target without a merge commit

Squash merge

Creates a linear history by condensing the source branch commits into a single new commit on the target branch

Rebase with merge commit

Creates a semi-linear history by replaying the source branch commits onto the target and then creating a merge commit

On

Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

Required

Block pull requests from being completed unless they have at least one linked work item.

Optional

Warn if there are no linked work items, but allow pull requests to be completed.

Off

Check for comment resolution

Check to see that all comments have been resolved on pull requests.

On

Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

Allowed merge types:

Basic merge (no fast-forward)

Preserves history exactly as it happened during development

Rebase and fast-forward

Creates a linear history by replaying the source branch commits onto the target without a merge commit

Squash merge

Creates a linear history by condensing the source branch commits into a single new commit on the target branch

Rebase with merge commit

Creates a semi-linear history by replaying the source branch commits onto the target and then creating a merge commit

- Crear al menos 2 branches de feature

- Realizar cambios y crear Pull Requests

Branch	Co...	Author	Authored...	Behind Ahead	Status	Pull ...
Feature						
Login	3d6f7e2	felipeganame	7m ago	0 0		
Register	3d6f7e2	felipeganame	7m ago	0 0		
main	3d6f7e2	felipeganame	7m ago			

New pull request

Feature/Login into main

Overview Files 1 Commits 1

Title

Updated README.md

Description

Updated README.md

17/4000

Markdown supported. Drag & drop, paste, or select files to insert.

Link work items.

@ # 🔗 📎 ↕ **B** *I* </> ↔ ☰ ☷ ☸

Updated README.md

Reviewers

Add required reviewers

Arnon Nahmias Search users and groups to add as reviewers

Work items to link

Search work items by ID or title

Updated README.md

Completed 12 AN Arnon Nahmias proposes to merge [Feature/Login](#) into [main](#)

[Overview](#) [Files](#) [Updates](#) [Commits](#)

Arnon Nahmias completed this pull request Just now

Cherry-pick

Revert

Merged PR 2: Updated README.md

4a93916e AN Arnon Nahmias Just now

[Show details](#)

✓ No required checks
Optional check succeeded

✓ Work items must be linked

✓ 1 reviewer approved

✓ No merge conflicts
Last checked Just now

Decisiones sobre Branch Policies en Azure Repos

En la rama principal **main** configuramos políticas para asegurar la calidad del código y controlar cómo se integran los cambios.

🔑 Configuraciones aplicadas

- **Require Pull Request:**

Todo cambio en `main` debe realizarse mediante un Pull Request, evitando que alguien suba código directamente sin revisión.

- **Minimum number of reviewers (1):**

Establecimos que al menos un integrante del equipo debe revisar cada PR. Esto garantiza colaboración y una validación mínima antes de integrar cambios.

- **Check for linked work items (Optional):**

Activamos la opción *Optional*. De esta forma, Azure DevOps recomienda vincular cada PR con un Work Item (User Story, Task o Bug), lo que mejora la trazabilidad, pero no bloquea el flujo de trabajo si el vínculo falta.

- **Limit merge types – Solo Squash merge:**

Permitimos únicamente *Squash merge*. Esto condensa todos los commits de la rama de feature en un único commit limpio dentro de `main`, manteniendo un historial ordenado y fácil de entender.

⚙️ Opciones que no usamos

- **Build Validation:**

No configuramos validaciones automáticas de compilación o tests porque la consigna no lo exigía. En un proyecto real se integrarían pipelines de CI/CD para validar cada PR.

- **Status Checks:**

No activamos verificaciones externas (ej. SonarCloud, GitHub Actions) porque exceden el alcance del trabajo. Sin embargo, en un entorno productivo son útiles para asegurar calidad de código y métricas.

- **Automatically included reviewers:**

No asignamos revisores automáticos porque el equipo es reducido. Preferimos asignarlos manualmente al crear cada PR. En un equipo grande esta opción sí sería recomendable.

Justificación

Con estas decisiones logramos un equilibrio entre cumplir la consigna y aplicar buenas prácticas:

- **Control de calidad mínimo garantizado** (PR + reviewer obligatorio).
- **Historial limpio** gracias al Squash merge.
- **Flexibilidad** en Work Items, sin bloquear innecesariamente.
- **Simplicidad** al no configurar integraciones que no eran requeridas.

Entregables

1. **Acceso al proyecto de Azure DevOps** con:
 - Todos los componentes configurados y funcionales
 - Historial de builds exitosos
 - Work items organizados en sprints
 - Pull Requests completados
2. **Repositorio en GitHub** con:
 - **README.md** con instrucciones para:
 - Acceder al proyecto de Azure DevOps
 - Clonar y trabajar con el repositorio
 - Ejecutar los pipelines
 - Entender la estructura del proyecto
 - **decisiones.md** explicando:
 - Metodología ágil elegida y justificación
 - Estructura de work items y por qué
 - **Evidencia de funcionamiento**: capturas mostrando:
 - Board con work items organizados
 - Pull Request aprobado y mergeado
 - Problemas encontrados y soluciones aplicadas
3. **URL del proyecto** en el Excel compartido:

-
<https://docs.google.com/spreadsheets/d/1mZKJ8FH390QHjwkABokh3Ys6kMOFZGzZJ3-kg5ziELc/edit?gid=0#gid=0>

🧠 Defensa Oral Obligatoria

Vas a tener que mostrar tu trabajo y responder preguntas como:

- ¿Cuál es la diferencia entre Epic, User Story y Task?
- ¿Cómo configuraste las políticas de branch y por qué?
- ¿Qué ventajas tiene usar Azure Pipelines sobre otras herramientas de CI/CD?
- ¿Qué es un Pull Request y cómo lo usaste?

✅ Evaluación

Criterio	Peso
-----	-----
Configuración técnica de Azure DevOps	25%
Claridad y justificación en `decisiones.md`	25%
Defensa oral: comprensión y argumentación	50%

⚠️ Uso de IA

Podés usar IA (ChatGPT, Copilot), pero **deberás declarar qué parte fue generada con IA** y justificar cómo la verificaste.
Si no podés defenderlo, **no se aprueba**.