1. Stack Elegido y estructura del repo

Stack elegido

Componente	Tecnología	Justificación
Frontend	Node.js + Express + HTML/CSS/JS	Simple, ligero, fácil de construir y deployar
Backend	Node.js + Express + SQLite	API REST completa, base de datos embebida sin configuración compleja
Pipeline	Azure DevOps YAML	Nativamente integrado con Azure, versionado como código
Agente	Self-Hosted en Windows	Control total sobre el entorno de build, cumple requisito del TP

Estructura del repositorio

```
TP4-Cervellini-Oliveto/
├── README.md                 # Documentación del proyecto
├── decisiones.md             # Este archivo con decisiones técnicas
├── azure-pipelines.yml       # Pipeline CI en YAML
├── .gitignore                # Exclusiones para Git
├── frontend/                 # Aplicación web (cliente)
│   ├── package.json          # Dependencias y scripts del frontend
│   ├── index.js              # Servidor Express para archivos estáticos
│   ├── index.html            # Interfaz de usuario principal
│   └── app.js                # Lógica del frontend (API calls)
└── backend/                  # API REST (servidor)
    ├── package.json          # Dependencias y scripts del backend
    └── index.js              # Servidor API con Express + SQLite
```

Endpoints de la API

- `GET /api/palabras` → Listar todas las palabras
- `POST /api/palabras` → Agregar nueva palabra
- `DELETE /api/palabras/:id` → Eliminar palabra por ID

2. Diseño del Pipeline

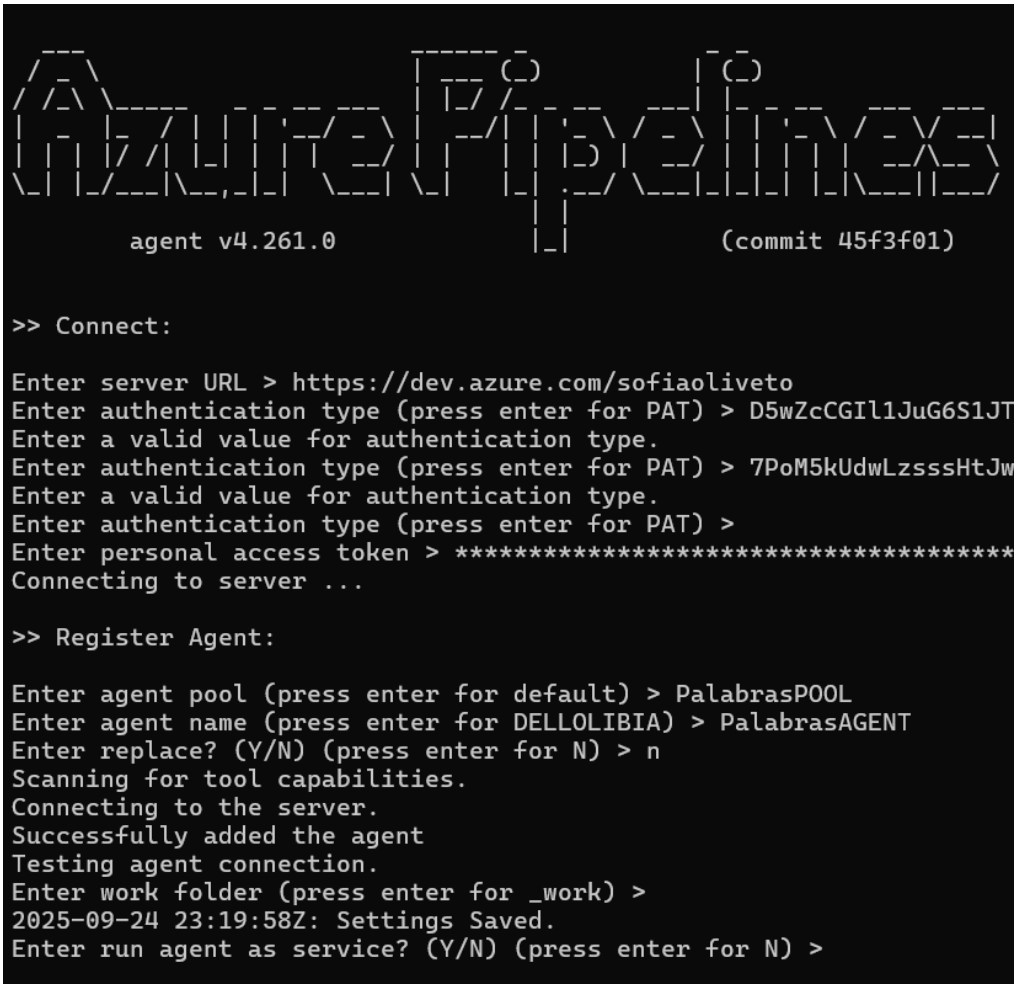
Para este trabajo se diseñó un **pipeline CI (Integración Continua)** utilizando YAML en Azure DevOps. El objetivo fue automatizar el proceso de **build y publicación de artefactos** de una aplicación compuesta por un frontend y un backend desarrollados en Node.js.

Estructura del pipeline

El pipeline está compuesto por:

- **Trigger**: Se ejecuta automáticamente ante cada push a la rama `main`.
- **Pool**: Utiliza un agente **Self-Hosted**, configurado en Azure DevOps bajo el nombre `PalabrasAGENT`, dentro del pool `PalabrasPOOL`.

Evidencias de cómo se configuró el AGENT del POOL:

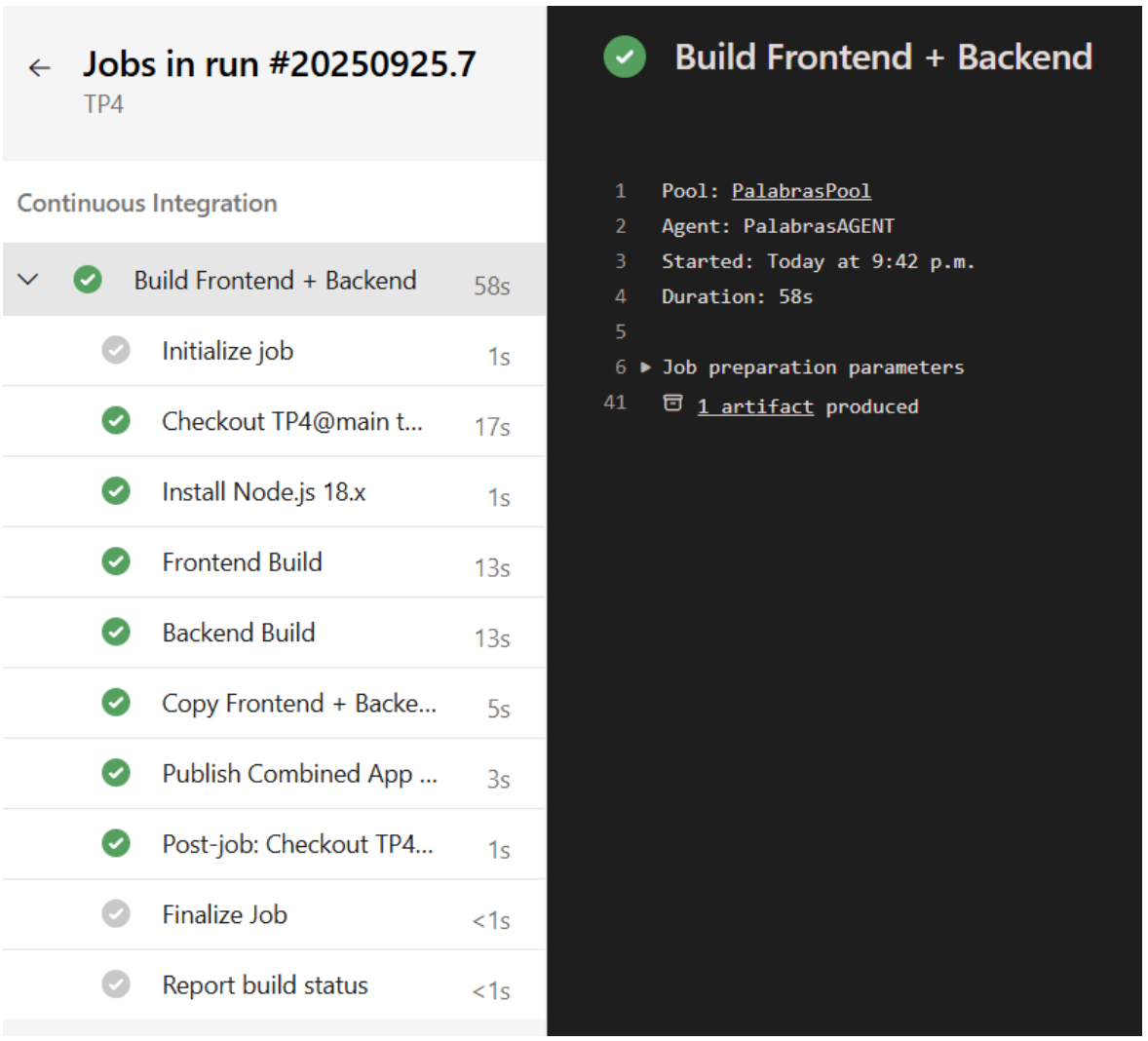


- **Stage**: Un único `stage` llamado `CI` (Continuous Integration).
- **Job**: Un único `job` `BuildApp`, que realiza todo el proceso de compilación.
- **Scripts**: Separan el build del front, el build del back y la publicación de artefactos

- **Artefactos**: Se generan artefactos combinados (frontend + backend) y se publican bajo el nombre `app-complete`.

Paso	Descripción
NodeTool@0	Instala Node.js 18.x en el agente
Build Frontend	`npm install` + `npm run build`
Build Backend	`npm install` + `npm run build`
Copy Frontend	Copia archivos a staging directory
Copy Backend	Copia archivos a staging directory
Publish Artifacts	Publica artefactos para uso posterior

Evidencias del build exitoso:



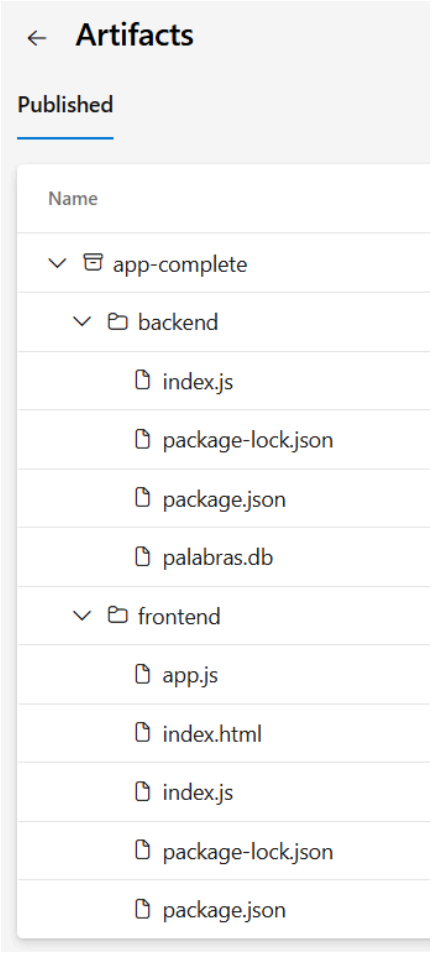
Y en el agente self-hosted:

```
2025-09-25 00:42:43Z: Running job: Build Frontend + Backend
2025-09-25 00:43:56Z: Job Build Frontend + Backend completed with result: Succeeded
2025-09-25 00:43:58Z: Running job: Build Frontend + Backend
2025-09-25 00:45:29Z: Job Build Frontend + Backend completed with result: Succeeded
```

Artefactos Generados

Artefacto	Contenido	Uso
frontend-artifact	`index.js`, `index.html`, `app.js`, `package.json`	Deploy del cliente web
backend-artifact	`index.js`, `package.json`	Deploy de la API REST

Evidencias de los artefactos publicados:



3. Problemas Resueltos Durante el Desarrollo

Primer problema

- **Error**: Error "`Cmd.exe exited with code '1'`"
- **Problema**: Pipeline fallaba con error de ejecución en Frontend y Backend
- **Causas múltiples**:
    - Variables apuntaban a carpetas inexistentes ('front', 'back')
    - npm ci requería package-lock.json que no existía
    - Scripts 'build' no definidos en package.json
- **Solución**: Pipeline simplificado con rutas directas y comandos flexibles
- **Resultado:** Build exitoso

Segundo problema

- **Error**: No había errores en el build, había advertencias: "`Directory is empty. Nothing will be added to build artifact`"
- **Problema**: No se publicaban artefactos
- **Causa**: Rutas incorrectas en CopyFiles@2
- **Solución**: Actualización del pipeline para usar nombres correctos
- **Resultado:** Build exitoso con publicación de artefactos