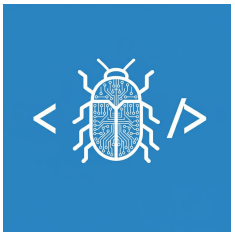

BugBusters

Come Usare il POC Nexum

Guida all'installazione e utilizzo

Stato	Approvato per RTB
Verificatore	Alberto Pignat
Redattori	Marco Piro, Marco Favero, Luca Slongo
Destinatari	BugBusters, Eggon, Prof. Tullio Vardanega, Prof. Riccardo Cardin

Registro delle Modifiche

Versione	Data	Descrizione	Redatto	Verificato	Approvato
1.0.0	31/01/2026	Sistemata sezione su Linux	-	-	Luca Slongo
0.1.0	31/01/2026	Sistemata sezione su Linux	-	Alberto Pignat	-
0.0.5	31/01/2026	Sistemata sezione su Linux	Luca Slongo	-	-
0.0.4	31/01/2026	Sistemata sezione su macOS	Marco Piro	-	-
0.0.3	31/01/2026	Aggiunta sezione installazione su MACOS e Linux	Marco Favero	-	-
0.0.2	31/12/2025	Migliorie alla guida	Alberto Autiero	-	-
0.0.1	29/12/2025	Prima stesura della guida all'utilizzo del POC Nexum	Alberto Autiero	-	-

Indice

1 Introduzione

Questa guida spiega in dettaglio come configurare, avviare e utilizzare il Proof of Concept (POC) Nexum. Seguire attentamente le istruzioni per garantire il corretto funzionamento di tutti i componenti.

2 Installazione - Windows

2.1 Ruby

1. Visitare https://rubyinstaller.org/downloads/
2. Scaricare **Ruby+Devkit** versione x64 (64-bit)
3. Eseguire il file di installazione
4. Durante l'installazione, selezionare tutte le opzioni predefinite
5. **Importante**: Installare MSYS2 e le componenti di sviluppo (DevKit)
6. Verificare l'installazione:

Listing 1: Verifica Ruby

```
ruby --version
```

7. Installare Bundler:

Listing 2: Installazione Bundler

```
gem install bundler
```

2.2 PostgreSQL

1. Visitare https://www.postgresql.org/download/windows/
2. Scaricare PostgreSQL Installer per Windows
3. Eseguire il file di installazione
4. Durante l'installazione:
   - Scegliere le componenti predefinite
   - **Importante**: Impostare una password per l'utente `postgres` (non dimenticarla!)
   - Lasciare la porta predefinita (5432)
5. Configurare l'avvio automatico (opzionale):

Listing 3: Avvio automatico PostgreSQL su Windows

```
Set-Service postgresql-x64-18 -StartupType Automatic
```

Nota: Sostituire il numero di versione se diverso.

2.3 Node.js

1. Aprire PowerShell e eseguire:

Listing 4: Installazione Node.js con winget

```
winget install OpenJS.NodeJS.LTS
```

2. Verificare l'installazione:

Listing 5: Verifica Node.js

```
node --version
npm --version
```

3. Installare le dipendenze del progetto (nella cartella frontend):

Listing 6: Dipendenze frontend su Windows

```
cd frontend
cmd /c "npm install"
```

2.4 Configurare PowerShell

Listing 7: Policy di esecuzione PowerShell

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
   CurrentUser
```

Nota: Confermare con `Y` o `S`.

3 Installazione - Linux e macOS

3.1 Ruby

- **macOS**:
  - *Se Homebrew non è installato, eseguire prima:*
    `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
  - Installazione Ruby:
    `brew install ruby`
- **Linux (Ubuntu/Debian)**:
  `sudo apt-get install ruby-full build-essential`
- **Linux (Fedora/RHEL)**:
  `sudo dnf install ruby ruby-devel`

Dopo l'installazione, **per macOS**, è necessario configurare il percorso:

Listing 8: Configurazione PATH Ruby su macOS

```
echo 'export PATH="/opt/homebrew/opt/ruby/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Successivamente, verificare l'installazione:

Listing 9: Verificare e completare Ruby

```
ruby --version
gem install bundler
```

3.2 PostgreSQL

macOS:

Listing 10: Installazione e configurazione PostgreSQL 18 su macOS

```
brew install postgresql@18

echo 'export PATH="/opt/homebrew/opt/postgresql@18/bin:$PATH"' >>
   ~/.zshrc
source ~/.zshrc

brew services start postgresql@18
postgres --version
```

Linux (Ubuntu/Debian):

Listing 11: PostgreSQL su Ubuntu/Debian

```
sudo apt-get install postgresql postgresql-contrib
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

Linux (Fedora/RHEL):

Listing 12: PostgreSQL su Fedora/RHEL

```
sudo dnf install postgresql postgresql-server
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

3.3 Node.js

macOS:

Listing 13: Node.js su macOS

```
brew install node
node -v
npm -v
```

Linux: Seguire https://nodejs.org/en/download/package-manager

Installare le dipendenze del progetto:

Listing 14: Dipendenze frontend su Linux/macOS

```
cd frontend
npm install
```

4 Configurazione Iniziale

4.1 Configurare le Variabili d'Ambiente

La configurazione del progetto avviene tramite il file `backend/.env`. Una copia di esempio è fornita in `backend/.env.example`.

4.1.1 Creare il file .env

Copiare `.env.example` e rinominarlo in `.env`:

Windows (PowerShell):

Listing 15: Creare .env su Windows

```
cd backend
Copy-Item .env.example .env
```

Linux/macOS:

Listing 16: Creare .env su Linux/macOS

```
cd backend
bundle install
cp .env.example .env
```

4.1.2 Completare le Credenziali AWS

Aprire il file `backend/.env` e compilare le credenziali AWS:

Listing 17: Sezione AWS in .env

```
AWS_ACCESS_KEY_ID=la_tua_access_key
AWS_SECRET_ACCESS_KEY=la_tua_secret_key
AWS_SESSION_TOKEN=la_tua_session_token
```

Dove trovarle: Accedere a https://eggon.awsapps.com/start/, selezionare il ruolo "Bedrock-Bugbuster" e copiare le credenziali temporanee.

4.1.3 Configurare il Database

Nel file `backend/.env`, impostare la password PostgreSQL:

Listing 18: Database in .env

```
DB_HOST=localhost
DB_PORT=5432
```

```
DB_USER=postgres
DB_PASSWORD=la_tua_password_postgresql
DB_NAME_DEVELOPMENT=nexum_poc_development
DB_NAME_TEST=nexum_poc_test
```

Nota: `DB_PASSWORD` deve essere la password dell'utente `postgres` impostata durante l'installazione di PostgreSQL.

4.2 Inizializzare il Database

Windows:

Listing 19: Inizializzare DB su Windows

```
# Avviare PostgreSQL, alternativamente ad aprire pgAdmin
Start-Service postgresql-x64-18

# Dalla root del progetto, eseguire
cd backend
bundle exec rails db:create db:migrate db:seed
```

Linux/macOS:

Listing 20: Inizializzare DB su Linux/macOS

```
cd backend
bundle exec rails db:create db:migrate db:seed
```

5 Avvio del POC

5.1 Windows - Con Script PowerShell

Eseguire il comando nella root del progetto:

Listing 21: Avviare il POC su Windows

```
.\avvia-poc.ps1
```

Questo script apre due finestre PowerShell: una per il Backend e una per il Frontend.

5.2 Linux/macOS - Manuale

5.2.1 Terminale 1: Backend

Listing 22: Backend su Linux/macOS

```
cd backend
bundle exec rails server
```

5.2.2 Terminale 2: Frontend

Listing 23: Frontend su Linux/macOS

```
cd frontend
npm start
```

6 URL Disponibili

Una volta avviati i servizi, avrai accesso a:

6.1 Applicazioni Principali

- **Frontend Angular**: http://localhost:4200
- **Backend Rails API**: http://localhost:3000

7 Script PowerShell Disponibili

Il progetto include diversi script PowerShell utili:

Script	Descrizione
`avvia-poc.ps1`	Avvia sia Backend che Frontend (apre due finestre)
`avvia-backend.ps1`	Avvia solo il Backend
`avvia-frontend.ps1`	Avvia solo il Frontend
`verifica-postgres.ps1`	Verifica la connessione a PostgreSQL (chiede password)

Tabella 1: Script disponibili nel progetto

8 Troubleshooting

8.1 Errore: "Problemi di permessi in fase di configurazione"

- **Windows**: Esegui PowerShell o il prompt dei comandi come amministratore.
- **macOS/Linux**: Evita di usare `sudo` con `brew` o `npm` per non corrompere i permessi. Se ricevi errori come "Permission denied" o "EACCES", riprendi il possesso della cartella corrente eseguendo:
  `sudo chown -R $USER .`

8.2 Errore: "Database non inizializzato"

Se all'avvio si presentano errori relativi a tabelle mancanti o database inesistente, eseguire manualmente il comando di inizializzazione (dalla cartella principale del progetto):

Listing 24: Inizializzazione manuale Database

```
cd backend && bundle exec rails db:create db:migrate db:seed
```

8.3 Errore: "Password PostgreSQL errata"

- Verifica che la password nel file `backend/.env` corrisponda a quella dell'utente `postgres` impostata durante l'installazione.
- **Windows**: Usa lo script `verifica-postgres.ps1` per testare la connessione.
- **macOS/Linux**: Per testare la connessione manualmente, apri il terminale e lancia:

```
    psql -h localhost -U postgres
```

  Se compare il prompt `postgres=#`, la connessione è riuscita (digita `\q` per uscire). Se richiede una password, prova a inserirne una vuota o quella che hai messo nel file `.env`.

8.4 Errore: "AWS credentials expired"

- Le credenziali AWS scadono dopo alcune ore
- Ottieni nuove credenziali dal portale AWS IAM Identity Center
- Aggiorna il file `backend/.env` con le nuove credenziali

8.5 Errore: "Could not find gem"

Se compare questo errore, significa che mancano alcune dipendenze Ruby. Eseguire i seguenti comandi per installarle:

Listing 25: Installazione dipendenze mancanti

```
cd backend
bundle install
```

8.6 Errore: "Port already in use"

- La porta 3000 (backend) o 4200 (frontend) è già in uso
- Chiudi l'applicazione che usa quella porta o modifica la porta nel file di configurazione

8.7 Il server non si avvia

- Verifica che PostgreSQL sia in esecuzione
- Controlla che le variabili AWS siano configurate nel file `.env`
- Controlla i log in `backend/log/development.log`

8.8 Warning VIPS

- I warning VIPS (riguardanti moduli opzionali per immagini) possono essere ignorati - non bloccano il funzionamento dell'applicazione.

8.9 Errore: "Script execution is disabled on this system"

(Solo per Windows)
Se ricevi un errore quando provi a eseguire gli script PowerShell:

Soluzione 1: Cambiare la policy di esecuzione (permanente) Per l'utente corrente eseguire il seguente comando:

Listing 26: Cambiare policy di esecuzione PowerShell

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
   CurrentUser
```

Questo comando permette l'esecuzione di script locali e richiede la firma solo per script scaricati da Internet.

Nota: Potrebbe essere richiesto di confermare digitando `S` (Sì) o `Y` (Yes).

9 Note Importanti

- **Database**: Il seed iniziale crea una Company con ID=1, alcuni Tone di esempio e una conversazione di esempio
- **Company ID**: La maggior parte delle API richiede `company_id=1` (la company di default)
- **Storage**: I file caricati vengono salvati localmente in `backend/storage/`
- **Log**: I log del backend sono disponibili in `backend/log/development.log`
- **Hot Reload**:
  - Rails: Si ricarica automaticamente quando modifichi i file Ruby
  - Angular: Si ricompila automaticamente quando modifichi i file TypeScript/HTML

10 Riavvio Dopo Modifiche

Dopo aver modificato:

- **File Ruby (modelli, controller, ecc.)**: Rails si ricarica automaticamente
- **File di configurazione Rails**: Riavviare il POC (Ctrl+C e rieseguire lo script di avvio)
- **File Angular**: Angular si ricompila automaticamente
- **File .env**: Riavviare il POC (Ctrl+C e rieseguire lo script di avvio)

11 Risorse Aggiuntive

- **Gemfile**: Lista delle dipendenze Ruby in `backend/Gemfile`
- **package.json**: Lista delle dipendenze Node.js in `frontend/package.json`