ENTREGA PROYECTO # 4

PARCHE LECTOR
CI/CD WITH GITHUB ACTIONS

Presentado por:

Nicolas Arciniegas

Juan Jose Alvarez Lozano

Julian Dario Colmenares Saenz

Julian Santiago Becerra Pulido

Sebastián Castañeda Garcia

Profesor:

Carlos Andres Sierra Virguez

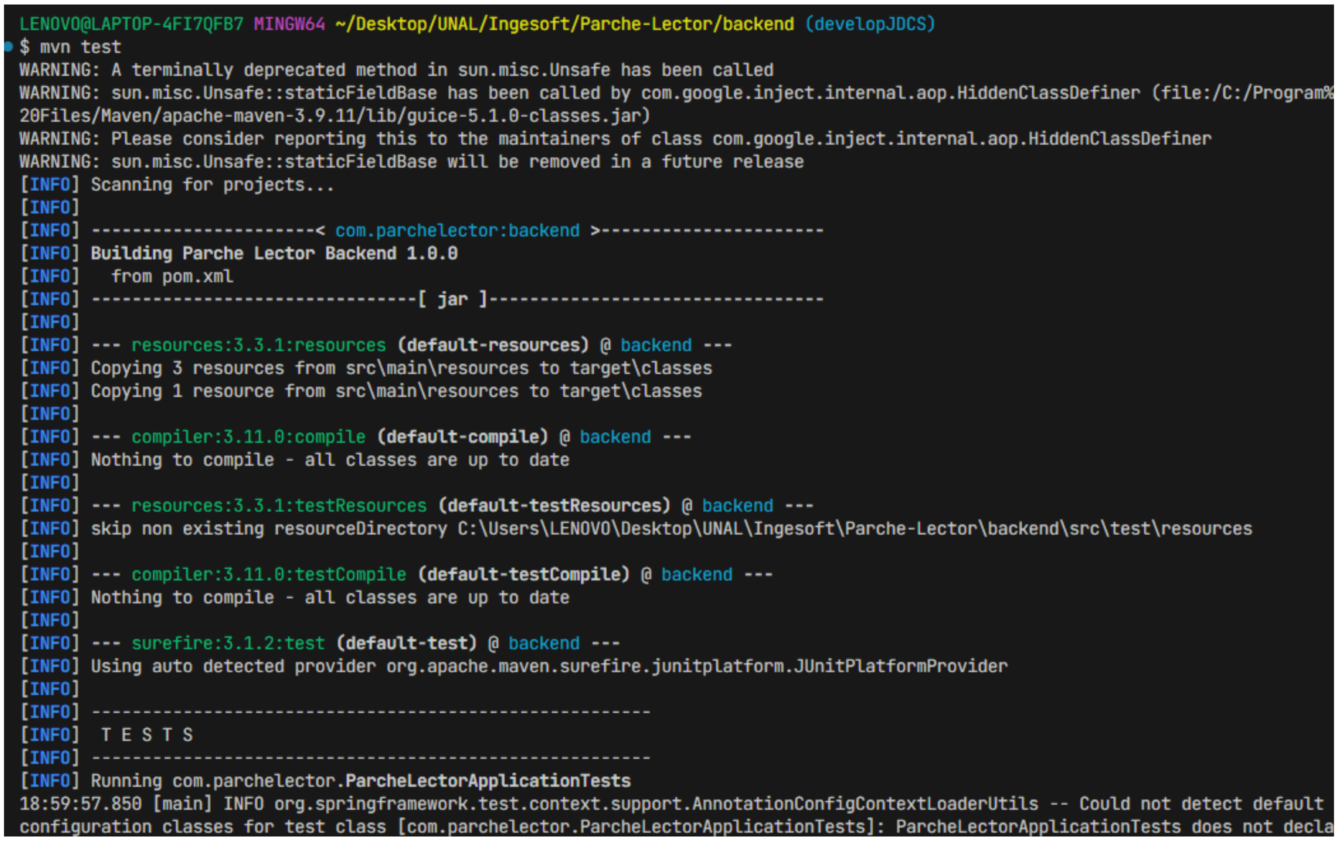
Domingo 30 de Noviembre

Universidad Nacional de Colombia
Departamento de Ingeniería de sistemas
2025

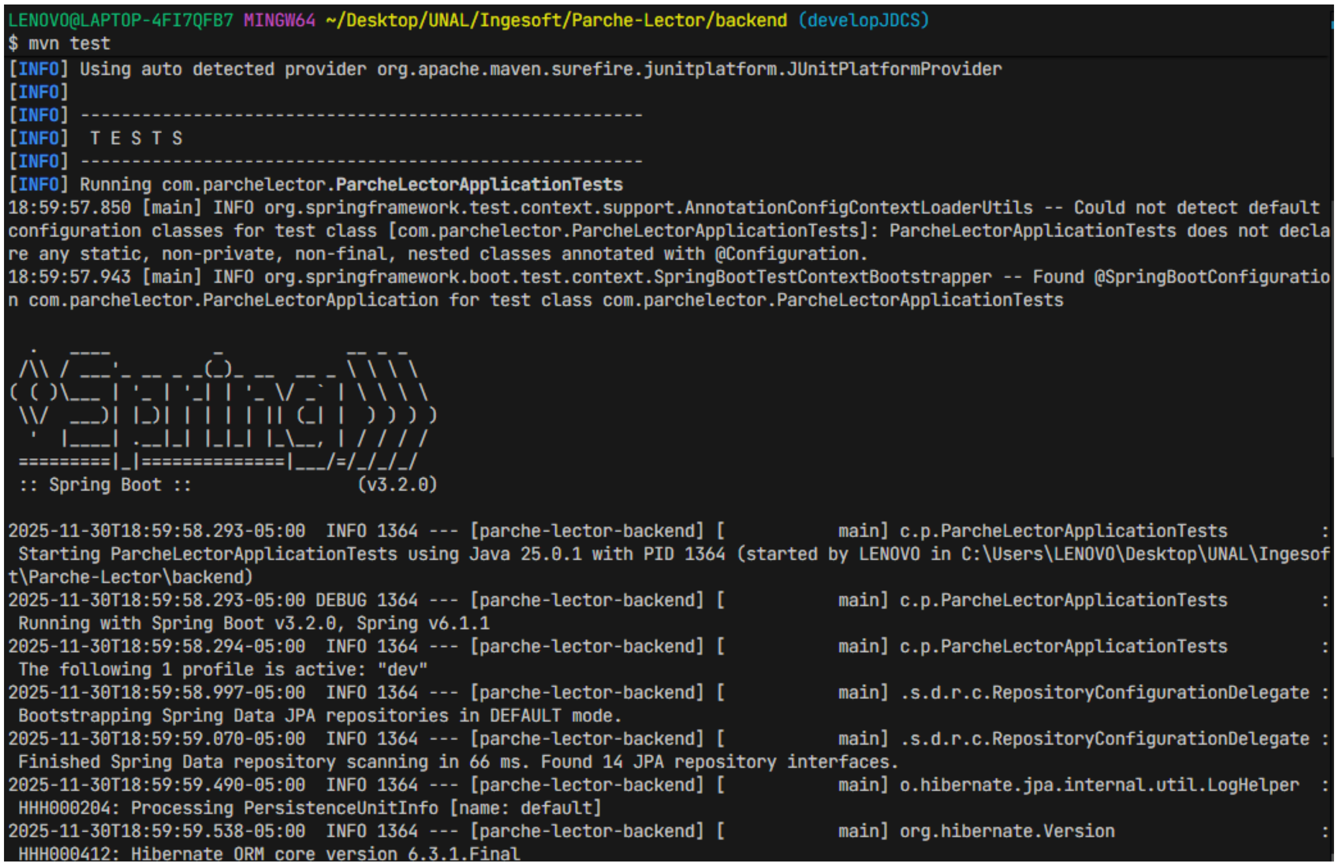
This document focuses exclusively on the Continuous Integration and Continuous Deployment (CI/CD) setup for the Parche Lector project. We describe the implemented GitHub Actions workflow used to automatically build the backend (Spring Boot) and frontend (Vue + Vite) Docker images on each push to the repository. The report includes step-by-step explanations of the pipeline, together with screenshots of successful workflow runs in GitHub Actions. In addition, we present basic automated tests executed locally for both the backend (Maven tests) and the frontend (Vitest), also documented with screenshots and brief explanations of how to reproduce each step.

Local execution of backend tests

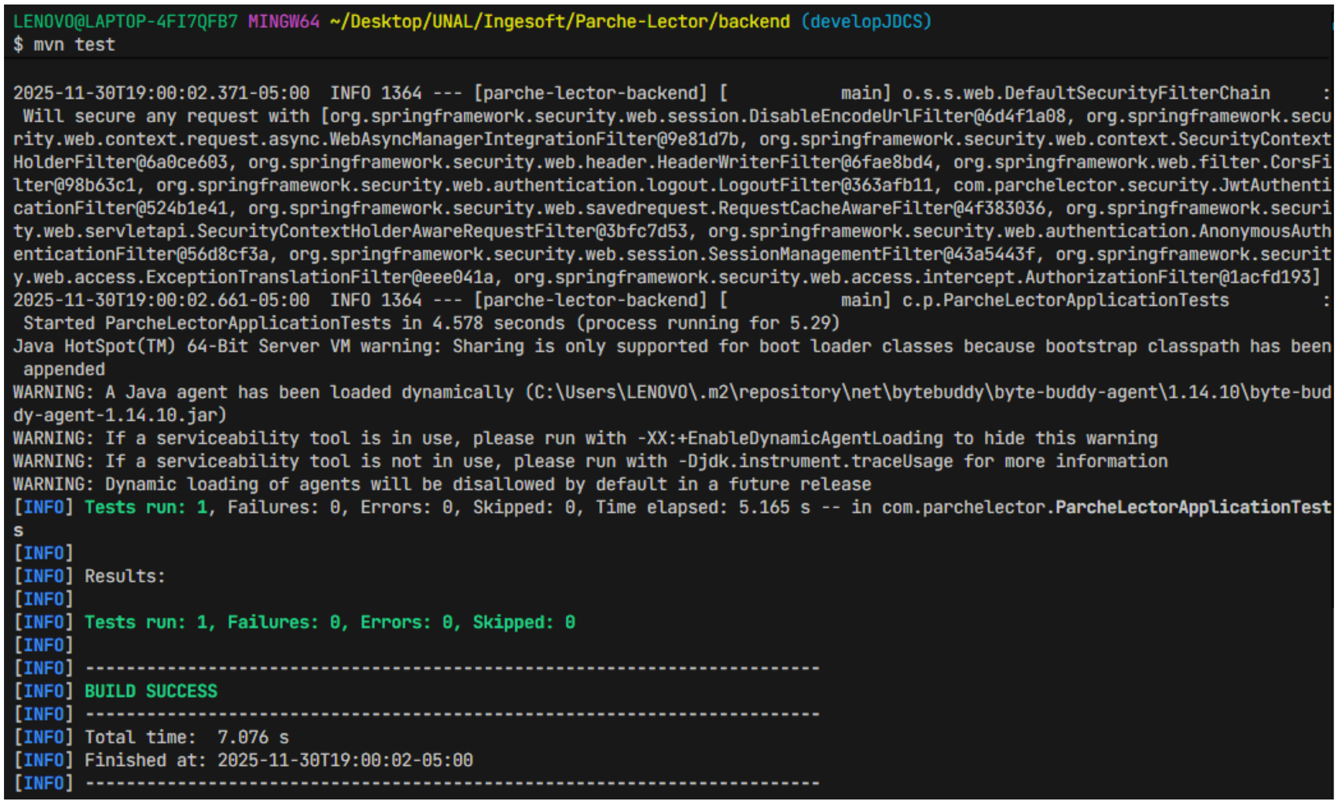
The backend tests were executed successfully using Maven, confirming that the Spring Boot application starts correctly and that the project structure allows proper context initialization. The generated output shows that the test suite ran without failures, validating that the backend environment is correctly configured for automated testing as part of the CI/CD pipeline. This ensures that future backend changes pushed to the repository will be automatically validated through GitHub Actions before integration or deployment.



```
LENOVO@LAPTOP-4FI7QFB7 MINGW64 ~/Desktop/UNAL/Ingesoft/Parche-Lector/backend (developJDCS)
$ mvn test
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::staticFieldBase has been called by com.google.inject.internal.aop.HiddenClassDefiner (file:/C:/Program%
20Files/Maven/apache-maven-3.9.11/lib/guice-5.1.0-classes.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.inject.internal.aop.HiddenClassDefiner
WARNING: sun.misc.Unsafe::staticFieldBase will be removed in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------< com.parchelector:backend >---------------------
[INFO] Building Parche Lector Backend 1.0.0
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]--------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ backend ---
[INFO] Copying 3 resources from src\main\resources to target\classes
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ backend ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ backend ---
[INFO] skip non existing resourceDirectory C:\Users\LENOVO\Desktop\UNAL\Ingesoft\Parche-Lector\backend\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ backend ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.1.2:test (default-test) @ backend ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.parchelector.ParcheLectorApplicationTests
18:59:57.850 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default
configuration classes for test class [com.parchelector.ParcheLectorApplicationTests]: ParcheLectorApplicationTests does not decla
```

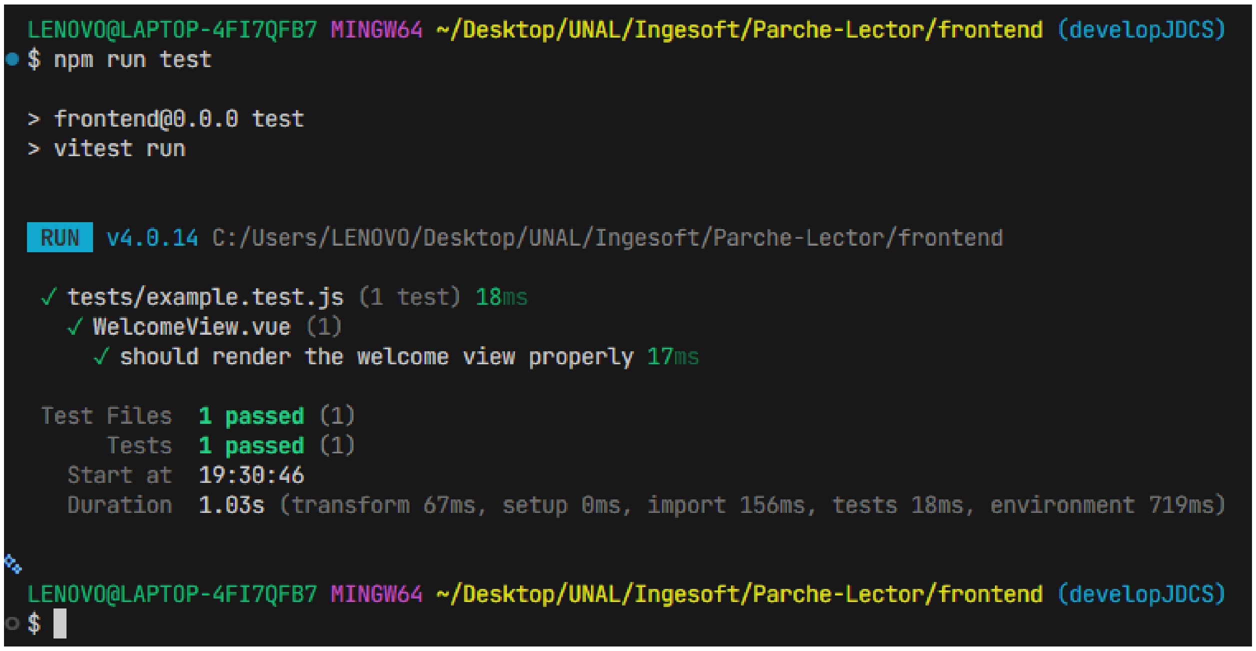


```
LENOVO@LAPTOP-4FI7QFB7 MINGW64 ~/Desktop/UNAL/Ingesoft/Parche-Lector/backend (developJDCS)
$ mvn test
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.parchelector.ParcheLectorApplicationTests
18:59:57.850 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default
configuration classes for test class [com.parchelector.ParcheLectorApplicationTests]: ParcheLectorApplicationTests does not decla
re any static, non-private, non-final, nested classes annotated with @Configuration.
18:59:57.943 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguratio
n com.parchelector.ParcheLectorApplication for test class com.parchelector.ParcheLectorApplicationTests

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.0)

2025-11-30T18:59:58.293-05:00  INFO 1364 --- [parche-lector-backend] [           main] c.p.ParcheLectorApplicationTests         :
 Starting ParcheLectorApplicationTests using Java 25.0.1 with PID 1364 (started by LENOVO in C:\Users\LENOVO\Desktop\UNAL\Ingesof
t\Parche-Lector\backend)
2025-11-30T18:59:58.293-05:00 DEBUG 1364 --- [parche-lector-backend] [           main] c.p.ParcheLectorApplicationTests         :
 Running with Spring Boot v3.2.0, Spring v6.1.1
2025-11-30T18:59:58.294-05:00  INFO 1364 --- [parche-lector-backend] [           main] c.p.ParcheLectorApplicationTests         :
 The following 1 profile is active: "dev"
2025-11-30T18:59:58.997-05:00  INFO 1364 --- [parche-lector-backend] [           main] .s.d.r.c.RepositoryConfigurationDelegate :
 Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-11-30T18:59:59.070-05:00  INFO 1364 --- [parche-lector-backend] [           main] .s.d.r.c.RepositoryConfigurationDelegate :
 Finished Spring Data repository scanning in 66 ms. Found 14 JPA repository interfaces.
2025-11-30T18:59:59.490-05:00  INFO 1364 --- [parche-lector-backend] [           main] o.hibernate.jpa.internal.util.LogHelper  :
 HHH000204: Processing PersistenceUnitInfo [name: default]
2025-11-30T18:59:59.538-05:00  INFO 1364 --- [parche-lector-backend] [           main] org.hibernate.Version                    :
 HHH000412: Hibernate ORM core version 6.3.1.Final
```



```
LENOVO@LAPTOP-4FI7QFB7 MINGW64 ~/Desktop/UNAL/Ingesoft/Parche-Lector/backend (developJDCS)
$ mvn test

2025-11-30T19:00:02.371-05:00  INFO 1364 --- [parche-lector-backend] [           main] o.s.s.web.DefaultSecurityFilterChain     :
 Will secure any request with [org.springframework.security.web.session.DisableEncodeUrlFilter@6d4f1a08, org.springframework.secu
rity.web.context.request.async.WebAsyncManagerIntegrationFilter@9e81d7b, org.springframework.security.web.context.SecurityContext
HolderFilter@6a0ce603, org.springframework.security.web.header.HeaderWriterFilter@6fae8bd4, org.springframework.web.filter.CorsFi
lter@98b63c1, org.springframework.security.web.authentication.logout.LogoutFilter@363afb11, com.parchelector.security.JwtAuthenti
cationFilter@524b1e41, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@4f383036, org.springframework.securi
ty.web.servletapi.SecurityContextHolderAwareRequestFilter@3bfc7d53, org.springframework.security.web.authentication.AnonymousAuth
enticationFilter@56d8cf3a, org.springframework.security.web.session.SessionManagementFilter@43a5443f, org.springframework.securit
y.web.access.ExceptionTranslationFilter@eee041a, org.springframework.security.web.access.intercept.AuthorizationFilter@1acfd193]
2025-11-30T19:00:02.661-05:00  INFO 1364 --- [parche-lector-backend] [           main] c.p.ParcheLectorApplicationTests         :
 Started ParcheLectorApplicationTests in 4.578 seconds (process running for 5.29)
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been
 appended
WARNING: A Java agent has been loaded dynamically (C:\Users\LENOVO\.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.10\byte-bud
dy-agent-1.14.10.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.165 s -- in com.parchelector.ParcheLectorApplicationTest
s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.076 s
[INFO] Finished at: 2025-11-30T19:00:02-05:00
[INFO] ------------------------------------------------------------------------
```

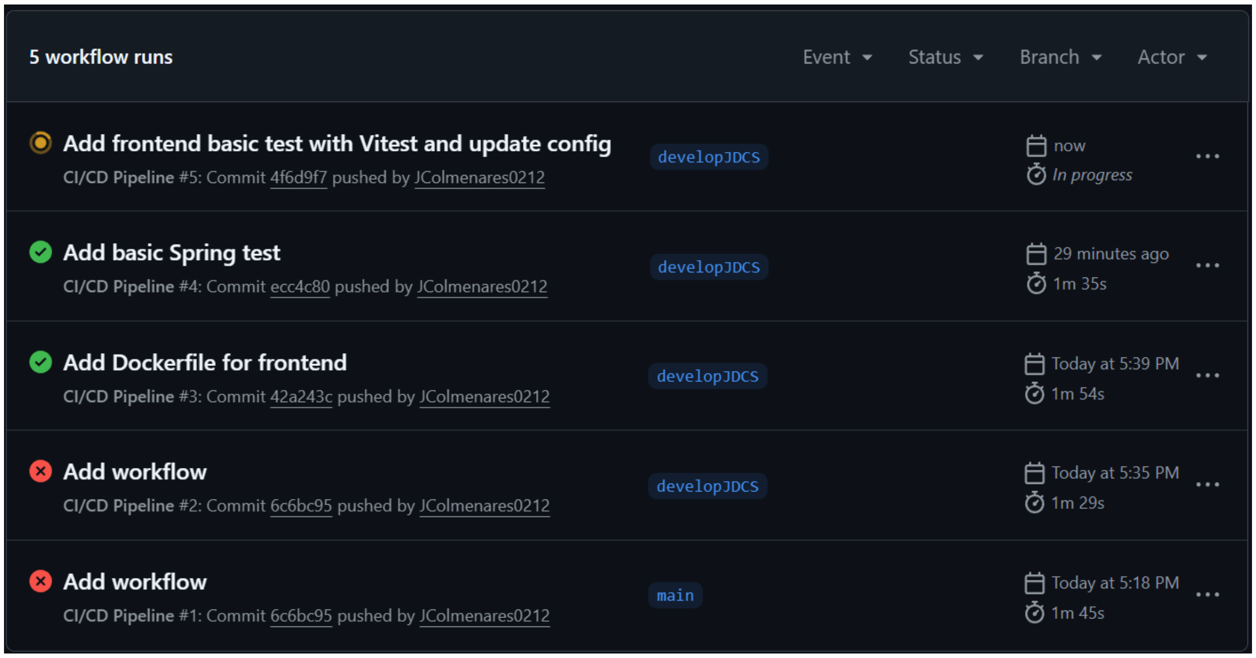
Local execution of frontend tests



```
LENOVO@LAPTOP-4FI7QFB7 MINGW64 ~/Desktop/UNAL/Ingesoft/Parche-Lector/frontend (developJDCS)
$ npm run test

> frontend@0.0.0 test
> vitest run


 RUN  v4.0.14 C:/Users/LENOVO/Desktop/UNAL/Ingesoft/Parche-Lector/frontend

 ✓ tests/example.test.js (1 test) 18ms
   ✓ WelcomeView.vue (1)
     ✓ should render the welcome view properly 17ms

 Test Files  1 passed (1)
      Tests  1 passed (1)
   Start at  19:30:46
   Duration  1.03s (transform 67ms, setup 0ms, import 156ms, tests 18ms, environment 719ms)


LENOVO@LAPTOP-4FI7QFB7 MINGW64 ~/Desktop/UNAL/Ingesoft/Parche-Lector/frontend (developJDCS)
$ 
```

The frontend test suite was executed locally using Vitest, and the results confirm that the testing environment is correctly configured. The test run completed successfully, with all test files and assertions passing without errors. This demonstrates that the project dependencies, test utilities, and component mounting are functioning as expected in the local development setup, validating that the frontend application can be reliably tested before integration into the CI/CD pipeline.

Pipeline in GitHub Actions


5 workflow runs

Add frontend basic test with Vitest and update config
CI/CD Pipeline #5: Commit 4f6d9f7 pushed by JColmenares0212
developJDCS
now
In progress

Add basic Spring test
CI/CD Pipeline #4: Commit ecc4c80 pushed by JColmenares0212
developJDCS
29 minutes ago
1m 35s

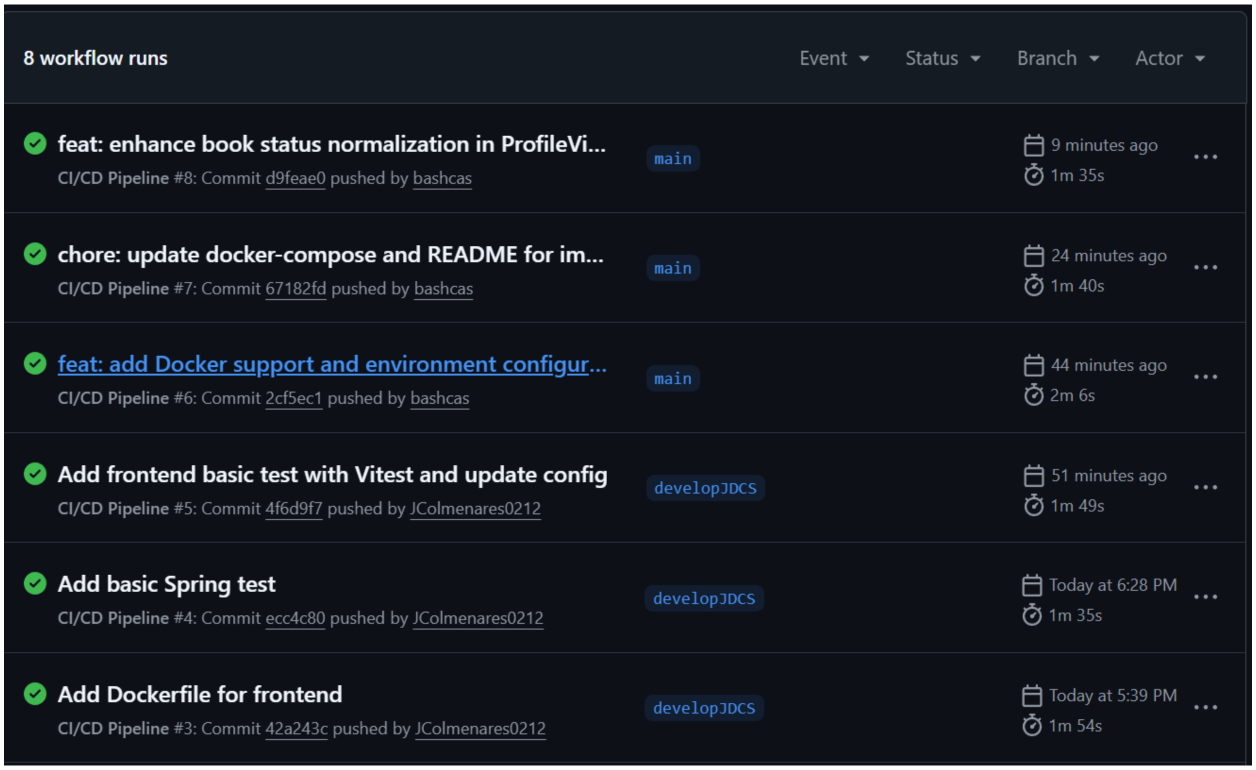
Add Dockerfile for frontend
CI/CD Pipeline #3: Commit 42a243c pushed by JColmenares0212
developJDCS
Today at 5:39 PM
1m 54s

Add workflow
CI/CD Pipeline #2: Commit 6c6bc95 pushed by JColmenares0212
developJDCS
Today at 5:35 PM
1m 29s

Add workflow
CI/CD Pipeline #1: Commit 6c6bc95 pushed by JColmenares0212
main
Today at 5:18 PM
1m 45s

The image shows the first series of CI/CD workflow runs in the project’s repository. These executions were performed intentionally on the developJDCS branch instead of main, since this branch was used as an isolated environment to configure and validate the pipeline before integrating it into the main development flow.

Each workflow corresponds to a specific incremental configuration step—such as adding the Dockerfile for the frontend, adding basic Spring Boot tests, and later configuring Vitest for the Vue frontend. The most relevant entry is the workflow titled “Add frontend basic test with Vitest and update config”, which demonstrates that the CI pipeline was successfully configured to automatically trigger a new workflow run on every push to the branch. This confirms that GitHub Actions is correctly detecting changes and executing the pipeline as expected, validating both Docker image builds and automated tests in an isolated development branch.


8 workflow runs

Event Status Branch Actor

feat: enhance book status normalization in ProfileVi...
CI/CD Pipeline #8: Commit d9feae0 pushed by bashcas — main — 9 minutes ago — 1m 35s

chore: update docker-compose and README for im...
CI/CD Pipeline #7: Commit 67182fd pushed by bashcas — main — 24 minutes ago — 1m 40s

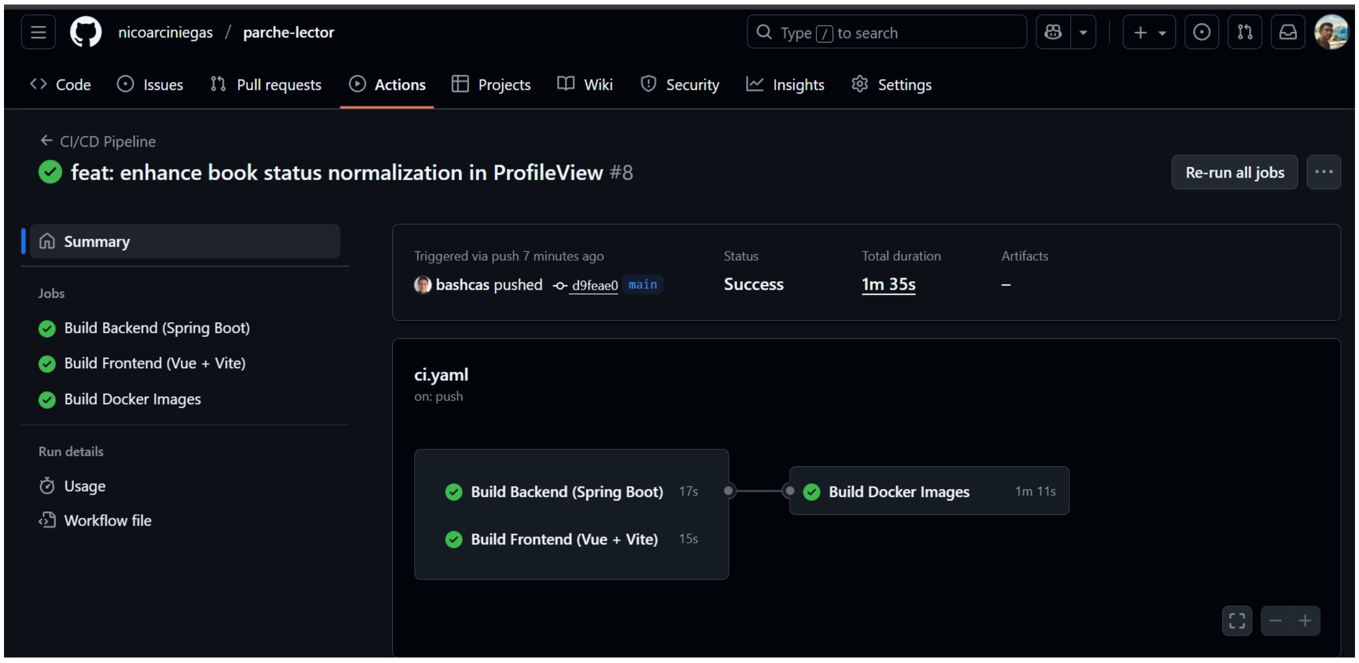
feat: add Docker support and environment configur...
CI/CD Pipeline #6: Commit 2cf5ec1 pushed by bashcas — main — 44 minutes ago — 2m 6s

Add frontend basic test with Vitest and update config
CI/CD Pipeline #5: Commit 4f6d9f7 pushed by JColmenares0212 — developJDCS — 51 minutes ago — 1m 49s

Add basic Spring test
CI/CD Pipeline #4: Commit ecc4c80 pushed by JColmenares0212 — developJDCS — Today at 6:28 PM — 1m 35s

Add Dockerfile for frontend
CI/CD Pipeline #3: Commit 42a243c pushed by JColmenares0212 — developJDCS — Today at 5:39 PM — 1m 54s

Some workflows working in the main branch


nicoarciniegas / parche-lector

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

CI/CD Pipeline
feat: enhance book status normalization in ProfileView #8

Re-run all jobs

Summary

Jobs
- Build Backend (Spring Boot)
- Build Frontend (Vue + Vite)
- Build Docker Images

Run details
- Usage
- Workflow file

Triggered via push 7 minutes ago — bashcas pushed d9feae0 main
Status: Success
Total duration: 1m 35s
Artifacts: –

ci.yaml
on: push

Build Backend (Spring Boot) 17s
Build Frontend (Vue + Vite) 15s
Build Docker Images 1m 11s

CI/CD workflow configuration

The file .github/workflows/ci.yml defines the complete CI/CD pipeline for the project. This workflow is automatically triggered on every push and pull_request to the repository (in our case we tested it mainly on the developJDCS branch). The pipeline is split into separate jobs:

Backend job (Spring Boot) – checks out the code, sets up Java and Maven, installs dependencies and runs mvn test to execute the basic Spring Boot context test.

Frontend job (Vue + Vite) – installs Node.js, runs npm install and then executes npm run test to run the Vitest suite for the frontend.

Docker images job – after the tests pass, it builds the backend and frontend Docker images using their respective Dockerfiles to ensure the code is ready to be containerized and deployed.

Thanks to this configuration, every new commit or pull request automatically goes through the same sequence of build, test and image-build steps, giving continuous feedback about the health of the application.

Dockerfiles for backend and frontend

Backend Dockerfile

The backend Dockerfile implements a two-stage build for the Spring Boot API.

In the first stage it uses a Maven + JDK base image to copy the pom.xml, download dependencies and compile the project with mvn clean package -DskipTests, producing a runnable JAR file. In the second stage it uses a lightweight JRE image, copies the generated JAR as app.jar, exposes the application port and defines the entrypoint java -Xmx350m -jar app.jar. This approach keeps the final image small and optimized for execution in production.

Frontend Dockerfile

The frontend Dockerfile is responsible for containerizing the Vue + Vite application. It uses a Node image to install dependencies (npm install) and run the production build (npm run build), generating static assets in the dist/ directory. Then, typically an Nginx (or similar minimal web server) image is used as the final stage, copying the compiled files into the web root and exposing port 80. This lets the frontend be served as a static, efficient SPA container that can be deployed alongside the backend container in the same docker-compose or deployment environment.