Report on the Deployment Process - StayHub

The complete deployment process for the frontend, database, and backend of the **StayHub** application is described below, highlighting the steps taken, the problems encountered, and their respective solutions.

1. Frontend Deploy (Angular + GitHub Pages)

The frontend deployment was carried out using **GitHub Pages**.

Initially, a manual configuration was attempted: running `ng build --prod`, uploading the generated files to the `docs/stayhub` folder, and pointing GitHub Pages to that folder.

However, upon accessing the URL, the application showed a blank page, and any route generated a **404 error**. The problem lay in the route handling of a **Single Page Application (SPA)** on GitHub Pages, which does not automatically redirect routes to the `index.html` file.

Implemented Solution:

- The `angular-cli-ghpages` tool was installed: `npm install -g angular-cli-ghpages`.
- The build was configured to generate files in the `docs/stayhub` folder
- The automatic deployment was executed: `ng deploy --base-href=/StayHub/`. This command:
    - Generated the optimized build in `docs/stayhub`.
    - Automatically created the `404.html` file to redirect all routes to `index.html`.
    - Uploaded the files directly to the `gh-pages` branch.
- In the repository settings on GitHub, **GitHub Pages** → `gh-pages` branch → `/docs/stayhub` folder was selected.

Result:

The application became available at: https://estebangmz666.github.io/StayHub/login

---

2. Database Deploy (PostgreSQL on Render)

Render was used to host a **PostgreSQL** instance on its free plan.

Steps Taken:

- On the Render dashboard, a new **PostgreSQL** instance was created:
    - Name: `stayhub-master`

    - Region: Oregon (USA)
    - Plan: Free
- Render automatically generated the credentials and connection URLs:
    - **Internal URL** (for internal use between Render services)
    - **External URL** (accessible from outside)

Initially, all credentials were copied to the project's local `.env` file. When attempting to connect from the backend, **it did not work** , as Render does not read `.env` files from the repository for security reasons.

Solution: The backend's **Web Service** on Render → **Environment** tab was accessed → **all environment variables were copied and pasted directly from the original .env**, including: `DATABASE_URL=${DATABASE_URL}` *Note: In this case, the **External URL** was the one that worked correctly, unlike the Internal URL, which did not allow the connection.*

After saving the changes, Render performed an automatic redeploy, and the connection was successfully established.

Result:

Operational database, accessible from the backend using the `DATABASE_URL` variable configured in Render.

---

3. Backend Deploy (Spring Boot + Render)

The backend, developed with **Spring Boot**, was deployed as a **Web Service** on Render, connected to the GitHub repository.

Problems Encountered and Solutions

Problem	Cause	Solution
`net::ERR_BLOCKED_BY_CLIENT`	Frontend on a different domain (GH Pages) trying to access the local backend	Configure CORS in Spring Boot with `@CrossOrigin` and deploy the backend first.
`Failed to convert String to Long` on `/request-password-reset`	Spring interpreted the path as a `@PathVariable` of type Long	"Change to `@PostMapping("api/v1/users/request-password-reset")` without parameters in the URL".

`Unexpected exception during bean creation (JWT)`	Environment variable was incorrectly named	Change `JWT_SECRET` → `jwt.secret.key` in Render's environment variables.
`Failed to configure a DataSource: 'url' attribute is not specified`	Spring could not find the DB URL	Use `application-prod.properties` + environment variables in Render (the production profile was maintained).
`Unable to determine Dialect`	Hibernate did not detect the PostgreSQL dialect	Add in `application-prod.properties`: `spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect`.
`Connection refused`	Incorrect use of Internal/External URL	Use the DB's **External URL** (worked better than the internal one in this case).
`"missing table [accommodation_amenities]"`	Hibernate did not automatically create the tables	Configure: `spring.jpa.hibernate.ddl-auto=update`.

Final Configuration on Render

- **Build Command**: `mvn clean package`
- **Start Command**: `java -jar target/stayhub-api-0.0.1.jar --spring.profiles.active=prod`
- **Environment Variables**: Copied from `.env` (including `jwt.secret.key`, `DATABASE_URL`, etc.)
- **Port**: Render assigns automatically (used by Tomcat on port 10000 internally)

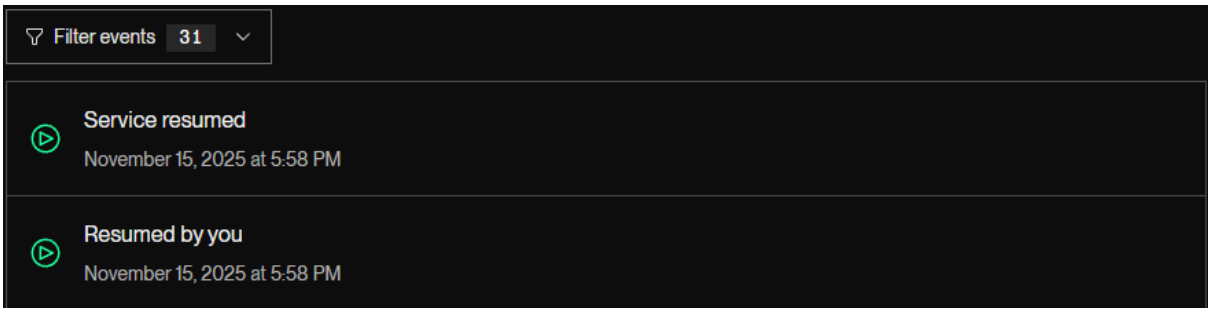
Result:

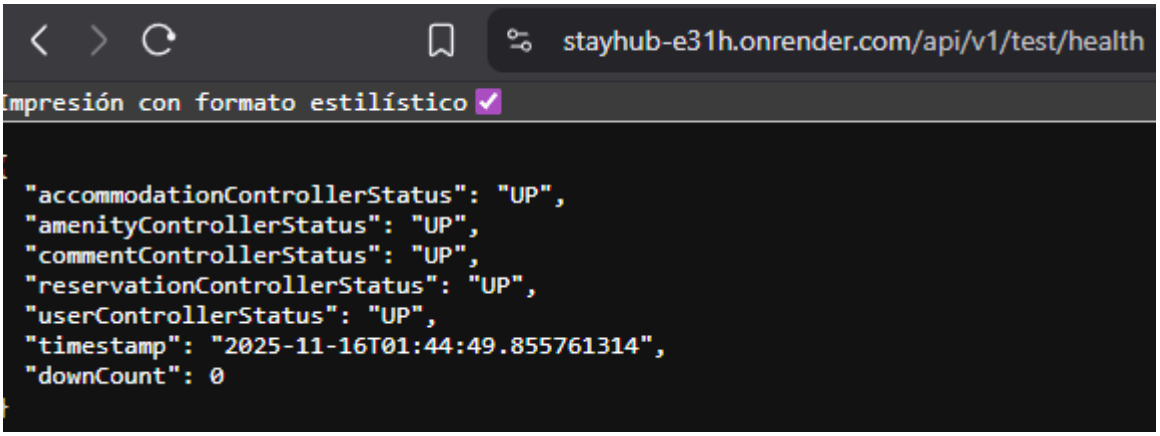
API deployed and operational at: `https://stayhub-e31h.onrender.com/`

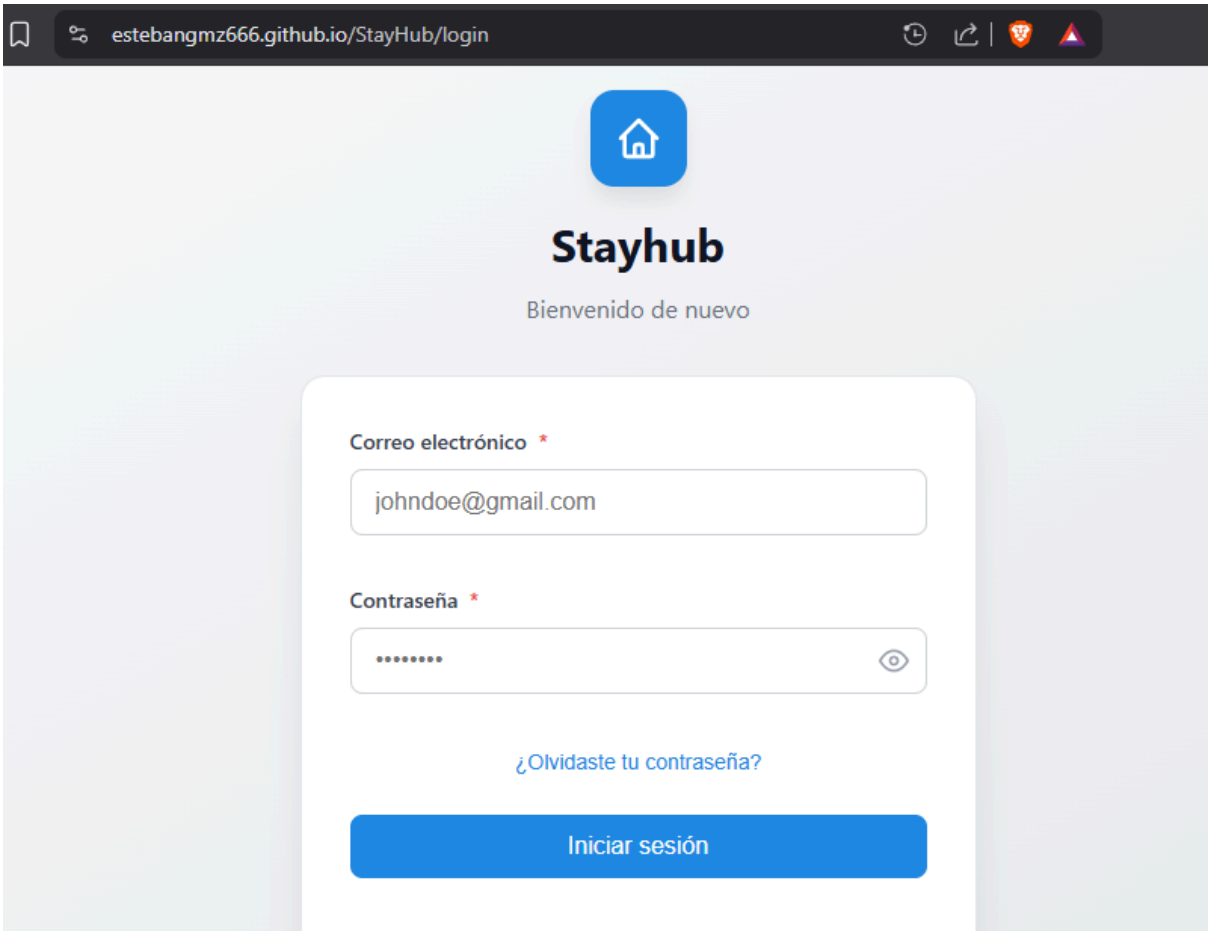
Final Logs:

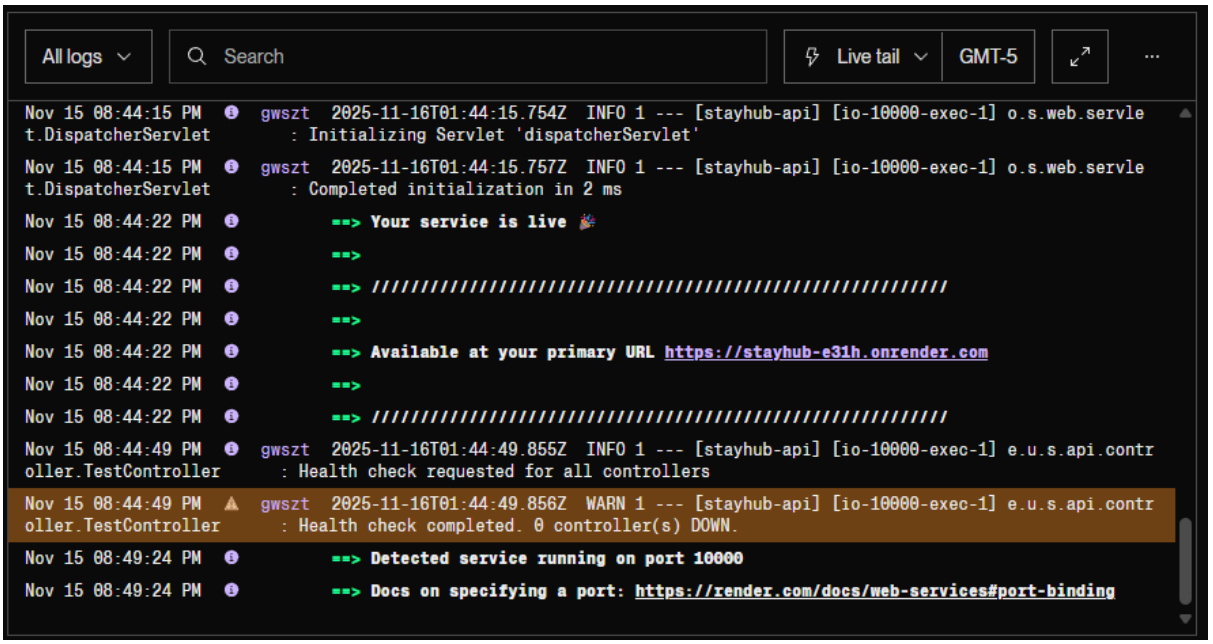
- `Tomcat started on port 10000`

- Database connected successfully
- Hibernate: table accommodation_amenities created
- Your service is live






estebangmz666.github.io/StayHub/login
Stayhub
Bienvenido de nuevo
Correo electrónico *
johndoe@gmail.com
Contraseña *
••••••••
¿Olvidaste tu contraseña?
Iniciar sesión


All logs
Search
Live tail
GMT-5
Nov 15 08:44:15 PM gwszt 2025-11-16T01:44:15.754Z INFO 1 --- [stayhub-api] [io-10000-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
Nov 15 08:44:15 PM gwszt 2025-11-16T01:44:15.757Z INFO 1 --- [stayhub-api] [io-10000-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
Nov 15 08:44:22 PM ==> Your service is live 🎉
Nov 15 08:44:22 PM ==>
Nov 15 08:44:22 PM ==> //
Nov 15 08:44:22 PM ==>
Nov 15 08:44:22 PM ==> Available at your primary URL https://stayhub-e31h.onrender.com
Nov 15 08:44:22 PM ==>
Nov 15 08:44:22 PM ==> //
Nov 15 08:44:49 PM gwszt 2025-11-16T01:44:49.855Z INFO 1 --- [stayhub-api] [io-10000-exec-1] e.u.s.api.controller.TestController : Health check requested for all controllers
Nov 15 08:44:49 PM gwszt 2025-11-16T01:44:49.856Z WARN 1 --- [stayhub-api] [io-10000-exec-1] e.u.s.api.controller.TestController : Health check completed. 0 controller(s) DOWN.
Nov 15 08:49:24 PM ==> Detected service running on port 10000
Nov 15 08:49:24 PM ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding

---

URL Summary

Component	URL / Detail
Frontend	https://estebangmz666.github.io/StayHub
Backend (API)	https://stayhub-e31h.onrender.com/
Database	PostgreSQL on Render (External URL configured in env vars)

Conclusions

The full-stack deployment required adjustments in each layer:

- **Frontend**: Correct handling of SPA routes.
- **Database**: Mandatory use of the environment variables panel.
- **Backend**: Precise configuration of profiles, variables, CORS, and DB connection.

Despite multiple configuration errors, all were resolved through log analysis, official documentation, and progressive adjustments. The **StayHub** application is now fully operational in production.