Final Project

Tópicos de Informática para Automação

Mário Antunes

December 12, 2025

Projects

This project is strictly individual. Select **one** of the following projects. All projects will be hosted on **GitHub**, using GitHub Classroom. Check here for details.

The repository must contain all relevant scripts, configuration files, and a `README.md` with instructions on how to deploy the project. It should also contain a project report in `PDF` format.

This is the **final project**, designed to integrate the various skills acquired throughout the semester (Shell Scripting, Docker, Python, Data Analysis, and Web Technologies).

This is a three-week project (deadline January 14, 2025). You have until the end of this week to notify your professor (via e-mail) of your chosen topic (the list of topics can be found here).

Do not forget to contact your professor with any questions. Further instructions may be added.

Topics

1. The IoT Simulator & Dashboard

- **Description:** Simulate a complete IoT pipeline using three Docker containers orchestrated by Compose.
  1. **Sensor Node (Python):** A script that generates synthetic "sensor data" (e.g., temperature, humidity, or energy usage) with random noise and sends it via HTTP POST every few seconds.
  2. **Collector API (FastAPI):** Receives the data, validates it, and appends it to a persistent log file (CSV or SQLite) stored in a **volume**.
  3. **Dashboard (Streamlit or Dash):** Reads the persistent data and displays a real-time auto-updating chart.
- **Core Topics:** Docker Compose, Python Scripting, APIs (FastAPI), Data Visualization, Volumes.

2. Automated System Monitor with Alerts

- **Description:** Create a robust system monitoring solution.
  1. **Agent (Bash Script):** A script running on the host (or a privileged container) that checks system health (Disk usage `df`, Memory `free`, or Load Average `uptime`). It should run periodically (loop or cron).
  2. **Logic:** If a threshold is breached (e.g., Disk > 90%), the script must trigger an alert.
  3. **Alert Service (Python):** A Dockerized Python script that accepts the alert message as an argument or environment variable and "logs" it to a structured JSON file with a timestamp.
  4. **Report:** Use **Pandas** to analyze the generated log file and print a summary of how many alerts occurred per day.
- **Core Topics:** Advanced Bash Scripting, System Commands, Python Data Processing, Error Handling.

3. The "Stock Market" Analyzer

- **Description:** Build a service that tracks and visualizes financial or crypto data.
  1. **Fetcher (Python):** A script that fetches real data from a public API (e.g., CoinGecko or a stock API) for 3 different assets over a specific period.
  2. **Storage:** Save this raw data into a structured CSV file.

3. **Web Server (Nginx + HTML/JS):** Serve a static webpage. Use **JavaScript (Fetch)** to load the CSV/JSON data and render a comparison chart using a library like **Chart.js** or **Leaflet** (if mapping locations).
4. **Analysis:** A separate Python script must calculate the "Volatility" (standard deviation) of the assets and output the riskiest asset to the console.

- **Core Topics:** External APIs, CSV handling, Web Server (Nginx), JavaScript/DOM, Statistical Analysis (Pandas).

4. The Secure Document Vault

- **Description:** Create a file storage system that demonstrates security and scripting.
  1. **Uploader (Python/FastAPI):** An API that accepts file uploads. Before saving, it must hash the file (SHA256) to ensure integrity.
  2. **Encryptor (Bash Script):** A script that watches the upload folder. When a new file arrives, it encrypts it using `gpg` or `openssl` with a predefined key/passphrase and deletes the original.
  3. **Access:** A simplified HTML frontend that lists the available (encrypted) files.
  4. The project report must explain the file permissions used (chmod/chown) to ensure only the script can read the raw files.
- **Core Topics:** Security concepts (Hashing/Encryption), Bash Automation, API File Handling, Linux Permissions.

5. The "Markdown Report" CI/CD Pipeline

- **Description:** Simulate a Continuous Integration pipeline for generating documentation.
  1. **Environment:** A Docker container containing **Pandoc**, **LaTeX**, and **Python**.
  2. **Trigger:** A **Bash script** monitors a folder for changes to a `data.csv` file.
  3. **Action:** When the data changes:
     - Run a Python script to generate new plots (PNGs) from the CSV.
     - Update a Markdown file to include the new date and stats.
     - Run **Pandoc** to compile the Markdown + Plots into a final PDF report.
- **Core Topics:** Dockerfiles (Custom Images), Bash Monitoring, Document Compilation, Python Plotting (Matplotlib).

6. Network Reconnaissance & Visualization

- **Description:** A tool to scan a network and visualize connected devices.
  1. **Scanner (Bash/Python):** A script that performs a ping sweep (or uses `nmap` if installed) on the local container subnet to discover active IP addresses.
  2. **Logger:** Store the active IPs and their response times in a database or CSV.
  3. **Visualizer (Python/Web):** A web service that displays the network topology (which IPs are up) and a histogram of response times.
  4. **Deployment:** Must be deployable via a single `docker-compose up` command that creates a private network for testing the scanner.
- **Core Topics:** Networking (Subnets/IPs), Docker Networking, Scripting, Data Visualization.

Github Classroom Access

Here are detailed instructions to access GitHub Classroom. Most students can skip several steps, given that these were completed in previous projects.

1. Join the Assignment and Form Your Team

1. **Access the link:** Go to here
2. **Find your name:** Select your name from the student list. > **Can't find your name?** All names registered on PACO were added. If yours is missing, please contact **Prof. Mário Antunes**.
3. **Create a team: Follow this exact naming structure (the nmec should be sorted): `[nmec]_project03`
   - *(Example:* `132745_project03`*)*

---

2. Access the Organization and Repository

1. **Accept the email invite:** After joining a team, all members will receive an email invitation to join the `detiuaveiro` GitHub organization.
2. **You must accept this invitation** before you can continue.
3. **Refresh the page:** Go back to the GitHub Classroom page and refresh it.
4. **Verify access:** You should now see and have access to your team's working repository.

---

3. Configure an SSH Key for Access

This will allow you to clone and push to the repository from your command line without entering your password every time.

1. **Check for an existing SSH key:** Open your terminal and run this command:

   ```
   cat ~/.ssh/id_ed25519.pub
   ```

2. **Generate a key (if needed):**
   - If you see a key (starting with `ssh-ed25519 ...`), copy the entire line and skip to step 3.
   - If you see an error like "No such file or directory," run the following command to create a new key:

     ```
     ssh-keygen -q -t ed25519 -N ''
     ```

   - After it's generated, run `cat ~/.ssh/id_ed25519.pub` again to view your new key and copy it.
3. **Add the key to your GitHub account:**
   - Go to your GitHub **Settings**.
   - On the left menu, click **SSH and GPG keys**.
   - Click the **New SSH key** button.
   - Give it a **Title** (e.g., "My UA Laptop").
   - Paste the key you copied into the **Key** field.
   - Make sure the "Key type" is set to **Authentication Key**.
   - Click **Add SSH key**.
4. **Authorize the key for SSO:**
   - After adding the key, find it in your list on the same page.
   - Click **Configure SSO**.
   - Select the **detiuaveiro** organization, fill in your login details, and grant access.