



---

Full Stack Developer  
**Technical Test**



## Development of an Application with own API and Frontend



**Test Objective:** To develop a web application for concert management, focused on managing live music events, including information about the concerts and the artists (groups or soloists) who will perform at them.

**Project Description:** The application will allow users to create, view, update and delete information about concerts and artists. Each concert can have one or more artists associated with it, and each artist (band or soloist) will have their own detailed information.

**Test delivery:** Send us the link to your repository (remember to make it public) and don't forget to fill in the README with all the necessary commands to deploy the project. For the evaluation, all the instructions in the README will be followed.



### Technical Requirements:

#### 1. Backend (API with Laravel):

- Develop a RESTful API to handle CRUD operations.
- Implement proper authentication and error handling.
- Use Laravel Jobs for scheduled or background tasks, if relevant to the application.

#### 2. Frontend (React preferably, otherwise you can use Livewire):

- Create an interactive user interface to interact with the API.
- Implement a grid list to display the data (GET).
- Develop forms and actions for create, read, update and delete operations.

#### 3. Database (MySQL):

- Design and configure a MySQL database to store the necessary information.
- Implement relationships between tables.
- Generate seeders to hold data when deploying the application.





#### 4. Version Control (Git):

- Use Git for project version control.
- Maintain an organised and descriptive commit history.

#### 5. Documentation:

- Create a README.md file with detailed instructions for deploying and running the project.
- Include technical and user documentation.

## Evaluation Criteria:

-  **Code Quality:** Organisation, readability and best practices.
-  **Functionality:** Compliance with CRUD operations requirements and effective communication between frontend and backend.
-  **Database Design:** Logical, efficient structure and adequate standardisation.
-  **Documentation:** Clarity and completeness in the README and additional documentation.



## Commits structure and their names:

1. In case of fire: git commit, git push and leave building:
  - Laravel skeleton creation and initial configuration.
  - Initial database configuration.
2. Delete without where:
  - Creating models and migrations for concerts and artists.
  - Definition of relationships between models.
3. Reinventing the wheel. Again!
  - Development of API endpoints for concerts and artists.
  - Implementation of authentication.
4. This commit is a lie:
  - Creation of Livewire components to list concerts and artists.
  - Initial user interface design.
5. In localhost works!
  - Implementation of forms for creating and editing concerts and artists.
  - User interface improvements and validations.
6. Let's f\*cking test!
  - Development of additional functionalities (if any).
  - Implementation of unit and/or integration tests.
7. DOG DOG DOG DOG!
  - Elaboration of technical and user documentation.
  - Deployment instructions in README.md.
8. Final BOSS:
  - Bug fixes, final adjustments.
  - Improvements based on feedback or testing.

We hope you enjoy doing this test and see you soon. **Best of luck!** 