We will evaluate:

- **Best practices:** Validations and dealing with exceptions, Intuitive names and OOP.
- **Solid principles:** Separation of concerns, abstraction and scalability, use of interfaces, inversion of dependencies.
- **Design patterns:** One or more design patterns are implemented to solve the main problem, choose the right channels and send notifications.
- **Architecture:** Good architecture design, well-defined folder structure and separation of concerns, scalable and prepared for minimal changes for new requirements in the future. (Routes/Controllers/Services/Repositories, DTOs/Interfaces/etc.).
- **Unit testing:** tests for each service and each function, multiple test scenarios per function.
- **Database:** migrations and seeders, use of foreign keys (in the case of RDBMS), indexing, correct data types and lengths and loading all catalogs into the database (is a plus).
- **Challenge:** fulfillment of requirements, performance and search methods, fault tolerance when sending notifications and scalability to add more notification channels.

Notification Test

It is required to create a system capable of receiving messages, which will have a category and the body of the message. These messages will need to be forwarded to the system's users, who will already be pre-populated. In addition to being subscribed to message categories, these users will have specified the channels through which they would like to be notified, such as SMS, Email or Push Notification.

With this configuration, users will only receive notifications of messages that fall within the categories they are subscribed to and through the channels they have specified.

There will be **three** message categories:

- **Sports**
- **Finance**
- **Movies**

And there will be **three types of notifications**, each requiring its own class to manage the sending logic independently:

- **SMS**
- **E-Mail**
- **Push Notification**

It is necessary to design the architecture for sending notifications through various channels. At a minimum, there should be one class for each channel, along with a strategy to select the appropriate channel. Real messages need not be sent using third-party services; the focus is on establishing a structure to implement the logic in the future.

Additionally, it is essential to store all the relevant information required to verify that the notification has been successfully delivered to the respective subscriber. This includes details such as the message type, notification type, user data, timestamp, and any other pertinent information.

No user administration is required, you can use a Mock of users in the source code, and they must have the following information:

- ID
- Name
- Email
- Phone number
- Subscribed [ ] here you need to list all the categories where the user is subscribed.
- Channels [ ] a list of the notification's channels (SMS | E-Mail | Push Notification).

As a user interface, you must display 2 main elements.

1. Submission form. A simple form to send the message that contains 2 fields:
    - Category. List of available categories.
    - Message. Text area, confirm that the message is not empty.
2. Log history. A list of all data records in the log, sorted from newest to oldest.