Challenge_Backend_South_African_Numbers

Please find an attached CSV file "South African Mobile Numbers". This file contains both correctly and incorrectly formed South African Mobile Numbers. Using any Open Source Framework and making use of Object Oriented Coding design principles.
Consume the provided file via any of the following means eg. upload / console call / API.
Test each number and check for correctness, attempt to correct incorrectly formed numbers and reject numbers that are invalid. (27831234567 is the correct format for this exercise).
Store the results appropriately to DB / Temporary File - as per your discretion.
Display results by the following means - divide the states as follows: a. Display acceptable numbers b. Corrected numbers + what was modified c. Incorrect numbers.
Use the following methods to display the data by whichever means you feel make best sense for a user to interpret, or machine in the case of an API. a. HTML b. File (single file or divided - your choice) c. API endpoint
Include a user form to test a single number and get a response with a success status/error message - if incorrect, why? And if self corrected, what was done.
Application should include basic Documentation (readme) & Unit Tests. Making use of existing libraries for the heavy lifting is permitted. If you do use a database, please include seeding files.

Conway's Game of Life

The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead, (or populated and unpopulated, respectively). Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with two or three neighbors survives.
- Any dead cell with three live neighbors becomes a live cell.
- All other live cells die in the next generation. Similarly, all other dead cells stay dead.

The initial pattern constitutes the seed of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a tick. Each generation is a pure function of the preceding one. The rules continue to be applied repeatedly to create further generations.

Sito web pasticceria

Realizzare il sito web di una pasticceria rispettando le seguenti specifiche.

1. La pasticceria vende dolci che hanno un *nome* ed un *prezzo*.
2. Ogni dolce è composto da una lista di *ingredienti*.
3. **Opzionale**: indicare di ogni ingrediente quantità e unità di misura.
4. La gestione della pasticceria è in mano a Luana e Maria che vogliono avere il proprio account per poter accedere all'area di backoffice tramite email e password.
5. Nell'area di backoffice si possono gestire (CRUD) i dolci e metterli in vendita con una certa disponibilità (esempio: 3 Torte paradiso in vendita).
6. **Opzionale**: permettere di caricare una o più foto del dolce.
7. I dolci in vendita invecchiano ed in base al tempo trascorso dalla loro messa in vendita hanno prezzi diversi: il primo giorno prezzo pieno, il secondo giorno costano l'80%, il terzo giorno il 20%.
   Il quarto giorno non sono commestibili e devono essere ritirati dalla vendita.
8. Realizzare una pagina vetrina dove tutti possono vedere la lista di dolci disponibili e il prezzo relativo.
9. **Opzionale**: realizzare la pagina del dettaglio del dolce (a sé stante o visualizzabile tramite overlay), in cui sono visualizzati gli ingredienti indicati dalla ricetta.

Realizzazione

Sii creativo e divertiti: non cerchiamo la perfezione, ci interessa capire come lavori. Massima libertà di espressione. Implementa la parte di backend/frontend con le tecnologie che preferisci:

- PHP: Symfony 5 + Doctrine ORM + twig (template engine se vuoi fare server side rendering); Laravel; Yii; RDBMS a scelta tra MySQL, Postgresql
- Javascript: NodeJs/Express + MongoDB
- C# .net Core
- Python: Django o Flask
- Javascript FE - Angular, React, Vue
- Go, Rust, Haskell, Clojure…
- CSS - Bootstrap, Bulma, …
- Qualsiasi altra soluzione purché funzionante e riproducibile.

Elementi di valutazione

Costituiranno elementi utili ad ottenere una valutazione positiva:

- Utilizzare un framework (JS, PHP o Python) rispetto all'utilizzo del solo linguaggio *vanilla*;
- Caricare l'elaborato in un server Git remoto pubblicamente accessibile (GitHub, GitLab, BitBucket o simili);
- Scrivere la documentazione dell'elaborato;
- Scrivere test unitari, funzionali o end-to end usando i framework opportuni (ad es. phpunit per PHP, cypress.js per Javascript, ecc.);
- Utilizzare Docker e/o Docker compose;
- Se si utilizzano template di siti web, dichiararlo apertamente 😊.