English

General objectives of the exercise

The exercise is divided into two parts: backend development and frontend development.

The main focus of backend development is to define an API endpoint that returns the contents of a CSV file in JSON format.

The main objective of the frontend development is the creation of a single-page application that shows, following a defined design, the data provided by the aforementioned API.

Result

The exercise must be delivered as source code, with compilation instructions if needed. The result of the application build are static files, to be served on a browser via a Web Server.

Backend Development

The exercise can be done in two ways:

1) Do you know and use **Serverless**?

    - Serverless framework (in order of preference)
        - **Serverless Framework, AWS SAM**
    - Programming language (in order of preference)
        - **Node.js**
        - **PHP**
        - **.NET**
    - Cloud services (AWS)
        - **Lambda, API Gateway**

2) Alternatively, you can do it this way

    - Programming language (in order of preference)
        - **Node.js**
            - Framework (in order of preference): **ExpressJS, Koa.js**
        - **PHP**
            - Framework (in order of preference): **CakePHP, Symfony, Laravel, Zend**
        - **.NET**
            - **.NET Core**
    - Web Server (in order of preference)
        - **Nginx**
        - **Apache**

        - **Node / PHP built-in web server**

- Specifications:
    - the business logic:
        - read the CSV as stream, avoiding the loading of the entire file
        - page the results in groups of 100
    - the API is an http route of the type: / api / flyers? Page = 1 & limit = 100

Frontend Development

- Programming languages and markup: Javascript, HTML, CSS (Sass)
- Ability to use one of the following Javascript frameworks / libraries: React, Backbone
- Ability to use one of the following UI libraries:
    - Material Design Lite: https: // getmdl. io /
    - Material UI (React): http s: //material-ui-next.com/
    - Semantic UI: https://semantic-ui.com/
    - Semantic UI React: https://react.semantic-ui.com/
    - Bootstrap: https://getbootstrap.com /
    - React Bootstrap: https://react-bootstrap.github.io/
- Specifications:
    - use the views of the chosen framework
    - client-side call to the API / api / flyers developed in the backend
    - management of empty state, loading state, error state
    - functionality “add to favorites" using the localStorage of the browser
    - responsive layout that follows the mockup attached
    - compatibility with versions of Internet Explorer> = 11

Attachments

- Flyers Data:
    - CSV: to be used in the development of the API Backend
- Mockup:
    - iPhone
    - iPhone - Open Menu
    - iPad

Flyers Data

The data in the CSV correspond to a small portion of ShopFully's flyer data.
In particular:

- id: id of the flyer
- title: title of the flyer
- start_date: start date of the flyer offer

- end_date: expiration date of the flyer offer (hint: you shouldn't show expired flyers)
- is_published: boolean flag (example: we do not show flyers that have an is_published value of 0)
- retailer: name of the reference retailer
- category: name of the flyer category

Suggestions

- You prefer to run both the backend and frontend parts, rather than completing just one perfectly
- Once chosen the framework, stick to its specifications to perform the exercise and based on the official documentation
- Write abstract and reusable functions, foreseeing a possible growth of the application
- Keep an eye on the performance
- Once you have chosen the UI framework, look at all its components (buttons , cards, etc.) and use all those that, in your opinion, are used to create the layout
- Use images in different sizes for the flyer cards, take inspiration from the flyers of the doveconveni.it home page

Nice to have

- The HTML code written or generated by the DOM is clean, semantically correct and compliant with the W3C rules
- The code is readable and documented
- L application follows the main rules and best-practices for accessibility
- If localStorage is not supported, use cookies as fallback
- The elements of the Flyer grid are shown in lazy loading as the user scrolls the page

Italian

Obiettivi generali dell'esercizio

L'esercizio si divide in due parti: sviluppo backend e sviluppo frontend.

Obiettivo principale dello sviluppo backend è la definizione di un endpoint API che restituisca il contenuto di un file CSV in formato JSON.

Obiettivo principale dello sviluppo frontend è la creazione di una single-page application che mostri, seguendo un design definito, i dati forniti dalla API di cui sopra.

Risultato

L'esercizio deve essere consegnato come codice sorgente, con istruzioni di "compilazione". Il risultato del build dell'applicazione sono dei file statici, da servire su browser tramite un Web Server.

Sviluppo Backend

L'esercizio può essere svolto in due modalità:

1) Conosci e usi tecnologie **Serverless**?

    - Framework serverless (in ordine di preferenza)
        - **Serverless Framework, AWS SAM**
    - Linguaggio di programmazione (in ordine di preferenza)
        - **Node.js**
        - **PHP**
        - **.NET**
    - Cloud services (AWS)
        - **Lambda, API Gateway**

2) Alternativamente, puoi svolgerlo così

    - Linguaggio di programmazione (in ordine di preferenza)
        - **Node.js**
            - Framework (in ordine di preferenza): **ExpressJS, Koa.js**
        - **PHP**
            - Framework (in ordine di preferenza): **CakePHP, Symfony, Laravel, Zend**
        - **.NET**
            - **.NET Core**
    - Web Server (in ordine di preferenza)
        - **Nginx**

        - **Apache**
        - **Node/PHP built-in web server**

- Specifiche:
    - la business logic:
        - legge il CSV come stream, evitando il caricamento dell'intero file
        - pagina i risultati a gruppi di 100
    - la API è una route http del tipo: /api/flyers?page=1&limit=100

Sviluppo Frontend

- Linguaggi di programmazione e markup: Javascript, HTML, CSS (Sass)
- Possibilità di usare uno dei seguenti framework / librerie Javascript: React, Backbone
- Possibilità di usare una delle seguenti librerie UI:
    - Material Design Lite: https://getmdl.io/
    - Material UI (React): https://material-ui-next.com/
    - Semantic UI: https://semantic-ui.com/
    - Semantic UI React: https://react.semantic-ui.com/
    - Bootstrap: https://getbootstrap.com/
    - React Bootstrap: https://react-bootstrap.github.io/
- Specifiche:
    - utilizzare le View del framework scelto
    - chiamata lato client alle API /api/flyers sviluppate in backend
    - gestione di empty state, loading state, error state
    - funzionalità di "salva nei preferiti" sfruttando il localStorage del browser
    - layout responsive che segua il mockup allegato
    - compatibilità con versioni di Internet Explorer >= 11

Allegati

- Flyers Data:
    - CSV: da usare nello sviluppo della API Backend
- Mockup:
    - iPhone
    - iPhone - Open Menu
    - iPad

Flyers Data

I dati nel CSV corrispondono a una piccola parte dei dati dei volantini di ShopFully.
In particolare:

- id: id del volantino
- title: titolo del volantino

- start_date: data di inizio dell'offerta del volantino
- end_date: data di scadenza dell'offerta del volantino (suggerimento: non si dovrebbero mostrare volantini scaduti)
- is_published: flag booleano (esempio: non mostriamo i volantini che abbiano un valore di is_published uguale a 0)
- retailer: nome del retailer di riferimento
- category: nome della categoria del volantino

Suggerimenti

- Preferisci eseguire entrambe le parti backend e frontend, piuttosto che completarne una sola "alla perfezione"
- Una volta scelto il framework, attieniti alle sue specifiche per eseguire l'esercizio e basati sulle documentazioni ufficiali
- Scrivi funzioni astratte e riutilizzabili, prevedendo una possibile crescita dell'applicazione
- Abbi un occhio di riguardo alle performance
- Una volta scelto il framework UI, guarda tutti suoi componenti (buttons, cards, etc.) e utilizza tutti quelli che, secondo te, servono alla realizzazione del layout
- Usa immagini di diverse dimensioni per le card dei volantini, prendi ispirazione dai volantini della home page di doveconviene.it

Nice to have

- Il codice HTML scritto o generato dal DOM è pulito, semanticamente corretto e conforme alle regole del W3C
- Il codice è leggibile e documentato
- L'applicazione segue le principali regole e best-practice per l'accessibilità
- Se il localStorage non è supportato, usa i cookies come fallback
- Gli elementi della griglia di Flyer vengono mostrati in lazy loading man mano che l'utente scrolla la pagina