ВЕБ-ОРІЄНТОВАНИЙ ДОДАТОК ДЛЯ ВІДСТЕЖЕННЯ ЛАБОРАТОРНИХ РОБІТ

“TASKER”

*Гайдуков Ростислав В’ячеславович*, Україна

Чернівецький національний університет імені Юрія Федьковича

Факультет математики та інформатики

[rostyslav.haidukov@live.com](mailto:rostyslav.haidukov@live.com)

В навчального процесі часто виникають багато труднощів пов’язаних з організаційними питаннями. Багато завдань і лабораторних робіт – викладачу важко стежити, що вже виконав студент, що ще ні і яке у нього завдання взагалі. Це призводить до великої кількість паперової роботи. Це не є добре, бо замість того, щоб думати про навчальний процес викладачу доводиться думати про те, хто які лабораторні здав/не здав. Так як ми живемо в епоху інформаційних технологій, то чому б не автоматизувати цей рутинний процес?

Велику популярність набувають веб-додатки, які надають досвід користувача близький до користування звичайними Desktop-додатками. Це не дивно, тому що веб технології активно розвиваються і залишають свої недоліки в минуле. Все більше додатків тепер можуть похизуватися web-версією, і в той же час існуючі веб-додатки переносять більше логіки на клієнтську сторону – на Front-End.

Було вирішити розробити такий додаток, так званий Single-Page application, тобто насправді у нас, на відміну від традиційних веб-сайті, є один невеликий html файл і багато JavaScript коду, який буде керувати всім функціоналом і динамічно міняти нашу розмітку.

Додаток дозволяє створити певний Workspace, долучити туди студентів і слідкувати за прогресом студентів у виконанні завдань. В якості дизайну обрав популярний зараз Google Material.

При розробці Single-Page додатків доводиться монолітний додаток розбивати на 2 окремі додатки: клієнтський і серверний. Спілкуватися вони будуть по REST. REST – такий архітектурний стиль обміну ресурсів між додатками, який базується на протоколі HTTP і накладає ряд обмежень:

Одним з обмежнеть REST є відсутність стану на сервері. Тому, стан додатку (дані про поточного користувача в системі, авторизаційні дані) зберігається на клієнті.Для цього я викристовую WebStorage API, що є частиною HTML5 і дозволяє зберігати дані на клієнті. Для авторизації використовується технологія JSON Web tokens. Це стандарт для створення токенів доступа, які підписуються секретний ключом, відправляються на клієнт. Клієнт повинен в кожний HTTP запит на захищений ресурс включати цей токен з HTTTP заголовок. Сервер повинен, маючи секретний ключ, перевіряти валідність токена.

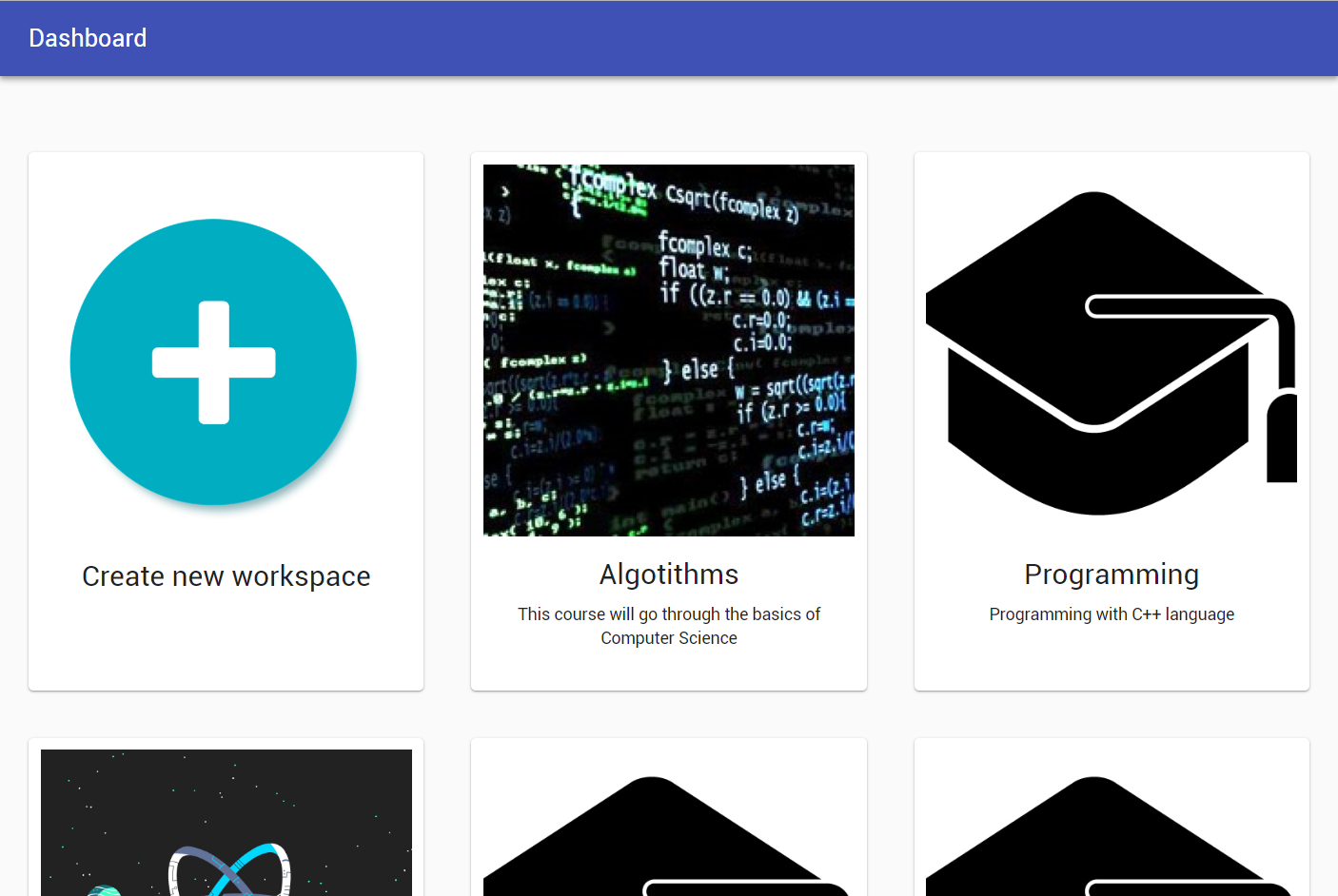
Для розробки Front-End використав добре відому бібліотеку React. Вона позволяє декларативно описувати складні інтерфейси веб та мобільних додатків. За рахунок використання технології Virtual DOM при будь-яких змінах в інтерфейсі перерендуватися буде тільки те, що змінилося, а не ціла сторінка. Це суттєве спрощує розробку додатків.

Якби ми не переносили логіку на клієнт, але нам все рівно потрібно десь зберігати дані, нам потрібне сервер з базою даних. В якості СКБД я обрав MongoDB. Це нереляційна документоорієнтована СКБД. Вона класифікує себе як NOSQL, використовує JSON-подібні документи для схеми даних. На відміну від традиційних СКБД, таких як MySQL чи PostgreSQL, тут не використовується табличний спосіб представлення і не пітримується мова запитів SQL. Основною перевагою є простота та гнучкість JSON формату, відсутність жорстко заданої схеми, що призводить до високої швидкості розробки.

Основним недоліком є обмежені можливості по забезпеченню консистентності даних, що може бути проблемою у додатках з великою кількістю відношень

Серверну частину (Back-End) реалізував на Node.js. Це середовище JavaScript, яке надає неблокуючу асинхронну модель для операцій вводу/виводу. В основі лежить движок JavaScript V8 (той самий, що використовується web-браузером Google Chrome) та C++ бібліотека libuv вона надає можливість з JavaScript коду звертатися до мережі та файлової системи, виконувати паралельні задачі тощо. В Node.js є event loop, який дозволяє за рахунок асинхронності виконувати багато операцій вводу/виводу.

Саме тому, Node.js добре себе показує при виконанні великої кількісті паралельниї, але не важких з обчислювальної точки зору задач.



<https://nodejs.org/uk/>

<https://reactjs.org/>

<https://developer.mozilla.org>

<https://www.mongodb.com/>

<https://webpack.js.org/>

<https://babeljs.io>

<https://expressjs.com/>

<https://mongoosejs.com/>

<https://en.wikipedia.org/wiki/Representational_state_transfer>

<https://jwt.io/>

<https://material.io/design/>