

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ОНУФРІЄВ ОЛЕКСІЙ СЕРГІЙОВИЧ

Допускається до захисту:

завідувач кафедри _____,

« _____ » _____ 20__ р.

Розробка інтернет-ресурсу для підтримки волонтерської діяльності

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:

Бабаков Р.М., _____

(назва кафедри)

Оцінка: ____ / ____ / _____

(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

(підпис)

АННОТАЦІЯ

Ця бакалаврська робота присвячена волонтерській діяльності та благодійних організаціях що шукають волонтерів, яка є досить актуальна з сучасним станом країни. Робота складається зі вступу, чотирьох розділів, висновку та додатку.

У вступі описана актуальність розробки інтернет-ресурсу для підтримки волонтерської діяльності та його використання для організацій та волонтерів.

У вступі ми розглянемо що таке волонтерство та навіщо воно потрібно

У першій частині ми розглянемо проблеми волонтерства, який стан волонтерства в Україні та розглянемо на зарубіжний досвід

У другій частині ми опишемо рішення проблем волонтерства і як веб додаток буде справлятися з цим

У третій частині ми опишемо алгоритми та підходи розробці

У четвертій та останній частині ми опишемо функції програми та як вони працюють

У висновку ми зведемо підсумок проведеної роботи та обґрунтуємо чому треба користатися цим додатком и які задачі він допоможе вам зробити.

ANNOTATION

This bachelor's thesis is dedicated to volunteering and charitable organizations looking for volunteers, which is quite relevant to the current state of the country. The work consists of an introduction, four chapters, a conclusion and an appendix.

The introduction describes the relevance of developing an Internet resource to support volunteering and its use for organizations and volunteers.

In the introduction, we will look at what volunteering is and why it is needed

In the first part we will consider the problems of volunteering, what is the state of volunteering in Ukraine and will consider the foreign experience

In the second part, we will describe the solution to the problems of volunteering and how the web application will cope with it

In the third part we will describe algorithms and approaches to development

In the fourth and last part we will describe the functions of the program and how they work

In conclusion, we will summarize the work done and justify why you need to use this application and what tasks it will help you do.

ЗМІСТ

Вступ.....	5
1 АНЛІЗ ВОЛОНТЕРСЬКОЇ ДІЯЛЬНОСТІ.....	6
1.1 Організації та волонтерська діяльність	6
1.2 Сучасні проблеми волонтерської діяльності	11
1.3 Постановка задачі	12
2 ОПИС ВИМОГ ДЛЯ СТВОРЕННЯ ВЕБ ДОДАТКУ	14
2.1 Функціональні вимоги візуального дизайну	14
2.2 Метод розробці.....	22
3 ШАБЛОНИ ПРОЕКТУВАННЯ ДОДАТКУ.....	24
4 РОЗРОБКА ДОДАТКУ	38
4.1 Технології з яких будуємо додаток.....	38
4.2 Архітектура додатку.....	40
4.3 Готовий продукт та тестування	43
ВИСНОВКИ	48
ПЕРЕДІК ДЖЕРЕЛ ПОСИЛАНЬ.....	49
ДОДАТОК.....	50

ВСТУП

Що таке волонтерство та навіщо воно потрібно? Волонтерство від латині voluntarius означає добровільний або добровольча діяльність. Ще це можна зрозуміти як наприклад, коли людина бачить проблеми біля себе та знає як їх виправити и це для не буде забирати багато ресурсів та часу він починає щось робити. З другої сторони є проблема яка потребує вирішення питань, це може бути будь яка праця, ремонт, будівництво чогось або підтримка людей що потрапили у складну ситуацію.

Коли з'явився чіткий рух волонтерства? Сама масово почали говорити про волонтерство це після битви при Сольферино 24 червня у 1859 року.

При якому були великі втрати, і ці втрати зауважив Швейцарський бізнесмен і письменник Анрі Дюнан, Дюана потрясла картина кровопролиття і людських страждань і він вирішив відкрити при місцевій церкві госпіталь за свій рахунок щоб лікувати постраждалих від війни, все люди приходили допомагати Дюан з госпіталем, жінки , діти, люди похилого віку і самі поранені допомагали йому безкоштовно.

Після повернення в Женеvu Дюнан написав книгу «Спогад про битву при Сольферіно» і вирішив зробити все можливе, щоб в подальшому зменшити страждання людей.

Книга швидко стала популярна, що за результатом в лютому 1863 року благодійна організація «Женевське суспільство благоденства» створила комітет з п'яти осіб, перед яким було поставлено завдання розглянути пропозиції Дюнана. Пізніше, в тому ж році, в Женеві була скликана міжнародна конференція, на якій і був заснований Червоний Хрест.

1 АНЛІЗ ВОЛОНТЕРСКОЇ ДІЯЛЬНОСТІ

1.1 Організації та волонтерська діяльність

У суспільстві завжди були виклики долі з якими держави не справлялися і в результаті таких подій з'являлися волонтерські організації.

Зараз з появою нових викликів в наш час, війною і ковід є багато проблем, які потребують вирішення але не встигають їх вирішувати. Так з'являються організації які в силі змінити поточну ситуація, часто це бувають такі ж співгромадяни які живуть в тому ж регіоні де трапилось лихо так само бувають і великі міжнародні організації які допомагають по всьому світу наприклад червоний хрест.

Ми розглянемо два типи організацій одну велику міжнародну а іншу маленьку локальну, які вони завдання вирішують і які у них проблеми є а потім ми спробуємо вирішити їх проблеми за допомогою нашого веб додатки, і так само можна порівняти результат як змінилися обставини у організаціях до і після. По дорозі ми розглянемо алгоритми, проектування, технології які ми використовували для розробки нашого продукту.

Одна з перших волонтерських організаціях був червоний хрест, він з'явився не просто так перед цим слідували страшні події, а саме в один з червневих днів 1859 відбулася битва між об'єднаними військами Франції проти армії Австрії на північній Італії, бої були довгі і з високими втратами, картина була жахлива, крім військових гинуло і мирне населення і допомоги не від якого боку не треба було, і в ці жахливі дні застав Анрі Дюнан швейцарський бізнесмен і письменник.

Дюнан був вражений жорстокої і безсмыслени війною, і був сильно вражений що немає ніякої допомоги. Обидві сторони кидали своїх військових на забій і кидали вмирати поранених, також страждало і мирне населення що отже допомоги не було де. Анрі організував госпіталь в місцевій церкві за свій рахунок і потім набирал в персонал всіх людей хто міг би допомагати, люди йшли на допомагати Анрі все, люди похилого віку, діти, жінки, ті ж поранені військові.

Всі важкі дні працював госпіталь, постраждалих постійно зростала. В кінці коли закінчилася битва і коли Анрі закінчив з роботою він повернувся в Женеvu і написав книгу про ці жахливі дні «Спогад про битву при Сольферіно» [1].

У книзі Дюнан описав все все жахливі дні з якими зіткнувся і твердив що такі події якщо не виходить уникати то їх потрібно мінімізувати, щоб постраждалих було якомога менше, і що потрібно зробити організацію яка буде допомагати в цьому рішенні.

Книга «Спогад про битву при Сольферіно» [1] швидко поширилася і шокувало публіку, і змусило людей переосмислити конфлікти, чи варті конфлікти людських жертв і чи можна їх уникнути, і чи можна мінімізувати втрати якщо конфлікти неминучі. Ці думки підштовхнули створити міжнародну організацію яка буде стежити за всіма конфліктами і допомагати постраждалим.

І в 1863 році благодійна організація «Женевське суспільство благоденства» вирішила створити комітет з 5 осіб яким довелося розглянути ідеї Анрі Дюнана. У тому ж 1863 році в Женеві була створена міжнародна конференція в якому створили міжнародну організацію Червоний Хрест скорочено МКЧХ, для прототипу прапора організації взяли швейцарський прапор і замінили кольори місцями, хрест з білого поміняли на червоний а фон з червоного на білий.

Коли МКЧХ утворився він себе оголосив про нейтральній стороні, біля якої стоїть завдання допомоги постраждалим від військових конфліктів, і у організації утворився принцип що у кожного конфлікту повинні бути рамки за які не можна виходити ні за яких обставин.

МКЧХ хоч і не є юридичною міжнародною організацією але має мандат недоторканності нарівні з ООН, що має недоторканність і імунітет. Що звільняється від податків, митних зборів, перевірок приміщень належить організації і від судового втручання.

МКЧХ позиціонується як нейтральна організація займається гуманітарними проблемами також організація була тричі удостоєна Нобелівської премії миру.

В Україні є представництво МКЧХ і вони працюють в Донбасі, вона допомагає людям по обидва боки хто потрапив в біду, і її імунітет дозволяє відвідувати окуповані території і рятувати людей від гуманітарної катастрофи. На окупованій території Донбасу вся інфраструктура зруйнована і знаходиться в занепаді а саме водопостачання повністю зруйнована, водопровід з початку конфлікту не ремонтували і немає коштів на це, МКЧХ поставляє обладнання і інструменти для організації «Вода Донбасу» яка працює на окупованій території і що її життєздатність сильно залежить від МКЧХ.

Так в активні роки конфлікту 2014 - 2015 де населення було відрізано від зовнішнього світу і не було коштів на існування і відсутності харчів що населення починало голодувати то МКЧХ постачав гуманітарну допомогу де були продукти харчування та питна вода що врятувало сотні тисяч життів.

Незважаючи на перемир'я на Донбасі бої тривають на прифронтових зонах що населення досі в гуманітарну катастрофу і МКЧХ продовжує поставляти продовольство і рятувати безліч життів по цей день.

У перший роки допомогою займалася також благодійний фонд «Розвиток України» та бойовики змусили її покинути окуповану територію а МКЧХ залишилася завдяки своєму міжнародному імунітету.

Тепер перемкнемося на локальні волонтерські організації і розглянемо її. Локальні організації створюються при локальних проблемах, на які не звертають уваги або не вистачає ресурсів від інших інститутів.

Їх і сама більше так як вони створюються і формуються швидко і допомагають вирішувати місцеві проблеми чітко. І одну з локальних організацій ми розглянемо «Відлуння».

«Відлуння» було сформована під крилом університету Українського Католицького Університету і перші волонтери були студенти його ж. Організація розташована у Львові і займається вирішенням проблем у Львові.

Як локальна організація вона знає всі проблеми в місті і кому потрібна допомога, швидко і чітко справляється з усіма викликами. «Відлуння»

займається різними проблемами наприклад як робота з дитячими будинками з собачими притулками і проблемами природи.

Розглянемо тепер роль волонтера, з його боку чому він зацікавлений допомагаючи іншим і як він змінює навколишнє середовище і як змінює себе разом з цим.

Якщо повернутися до Анрі Дюнану то допомагаючи потерпілим він себе переглянув свою діяльність і більш продуктивним став написавши відому книгу що змінило світогляд багатьох інших людей.

Відштовхуючись від цього ми можемо зрозуміти що допомагаючи іншим ми допомагаємо і собі, ми переглядаємо своє буття і і знаходимо чітке розуміння що робити в своєму житті, ми можемо порівняти своє життя з людьми потрапили в скрутну ситуацію і розуміємо що світ буває і по другому.

Ця колоритність і різнобічність прокачує наш світогляд. І ми стаємо більш усвідомленими. Так само допомагаючи іншим ми можемо побувати в інших місцях де живуть інші люди з іншими укладами, з іншою культурою і з іншими поняттями, де ми можемо взяти шматочок їх досвіду і культури і принести до себе додому, отже так само і ми робимо, ми приносимо свою культуру свій досвід і тим самим допомагаємо їм теж, що створюються міжкультурне порозуміння.

Розглянемо як розвиваються провінції і цілі регіони де залишають люди і перебираються до великих міст і як волонтерства допомагає залишитися в регіонах і не переїжджати до великих міст.

Наприклад в багатьох регіонах Бельгії є свої волонтерські організації, вони шукають волонтерів з всього світу на рік, за цей рік волонтери що приїхали займаються різними справами, будують логера для мігрантів, реставрують старі будинки та замки, допомагають робити івенти для локальної ком'юніті в маленьких селищах та саме головне приносять досвід зі своїх країн. Також після волонтерського року що люди проводять у Бельгії вони забирають новий досвід та привозять до себе до дому, що всі виграють в цій ситуації.

Також на бельгійському прикладі ми можемо побачити як вирішуються проблеми з маленькими селищами і як їх треба розвивати.

Та всі процеси у Бельгії вистроєні за допомогу веб додатку щодо волонтерської діяльності. Людей шукають через веб додаток, волонтери

заповнюють анкету свою та вибирають де и що вони хотіли би робити, а потім організація робить відбір та робить співбесіди с кандидатами.

Головний веб додаток що використовують у Бельгії це Compagnons batisseurs [4]. На прикладі бельгійської селища Марш-ан-Фамен можемо зробити деякі висновки, місцева влада виділила будинок для місцевого офіса де працюють волонтери та виділила будинок для житла. Та сюди приїжджать люди з всього світу то домагаються селищу розвиватися, підтримують інвалідні центри, місцевій дом культури та і просто роблять соціальну роботу що дуже допомагає місцевості.



1.2 Сучасні проблеми волонтерської діяльності

На сьогоднішній день у волонтерських організаціях залишаються проблемою в знаходженні волонтерів і проблема допоміжних інструментів які будуть допомагати менеджерам всі процеси їх робіт, від менеджменту волонтерів до побудов завдань для волонтерів та відстежень виконання їх, також є труднощі із зберіганням даних учасників і фільтрація їх до подальших робіт . Також є проблеми з інструментами які б заохочували волонтерську діяльність.

Тут ми прийшли до висновку що потрібен веб додаток яке б допомагало б і спрощувало роботу організацій, адже ми помітили як скільки багато завдань стоїть які другі не вирішують, і важливий фактор на швидкості допомоги та комунікації.

Важливий фактор для всіх робіт це структура, порядок та зберігання інформації. Нам потрібно спрощувати та покращувати роботу, щоб швидше і якісніше приходила допомогу, адже кожен хвилина буде коштувати дуже дорогих життів.

1.3 Постановка задачі

Наше рішення буде в веб додатку, що буде спрощувати і допомагати організаціям. Ми зробимо функції які в яких вони потребують а саме створимо всі важливі інструменти які будуть допомагати в повсякденних буднях.

А саме легкість комунікація, створення місій і відстеження виконують їх. Також важливий фактор буде доступ до бази волонтерів і легкість знаходження їх.

Так само зробимо напрямок волонтерства, щоб організації могли створювати місії для вузьких напрямків і знаходження волонтерів під певні критерії.

Так само потрібно розуміти що у нас є два типи користувачів це організації та волонтери.

Перші виступають за організаторів і займаються створенням місій що для них потрібно зручні інструменти для управління. Для них веб додатки повинен позиціонуватися як інструмент в роботі який буде спрощувати роботу і прискорювати всі процеси.

Другі виступають волонтерами тобто, люди у яких є час і ресурси які готові допомогти нужденним, для них ми позиціонуємо легкість і комунікацію з новими людьми, також отримання нового досвіду, обмін культури і гарного проведення часу.

Так само потрібен простий і тямущий дизайн де ми вході в веб додатки ставало відразу все зрозуміло, де що знаходиться і для чого, як воно працює і який буде результат.

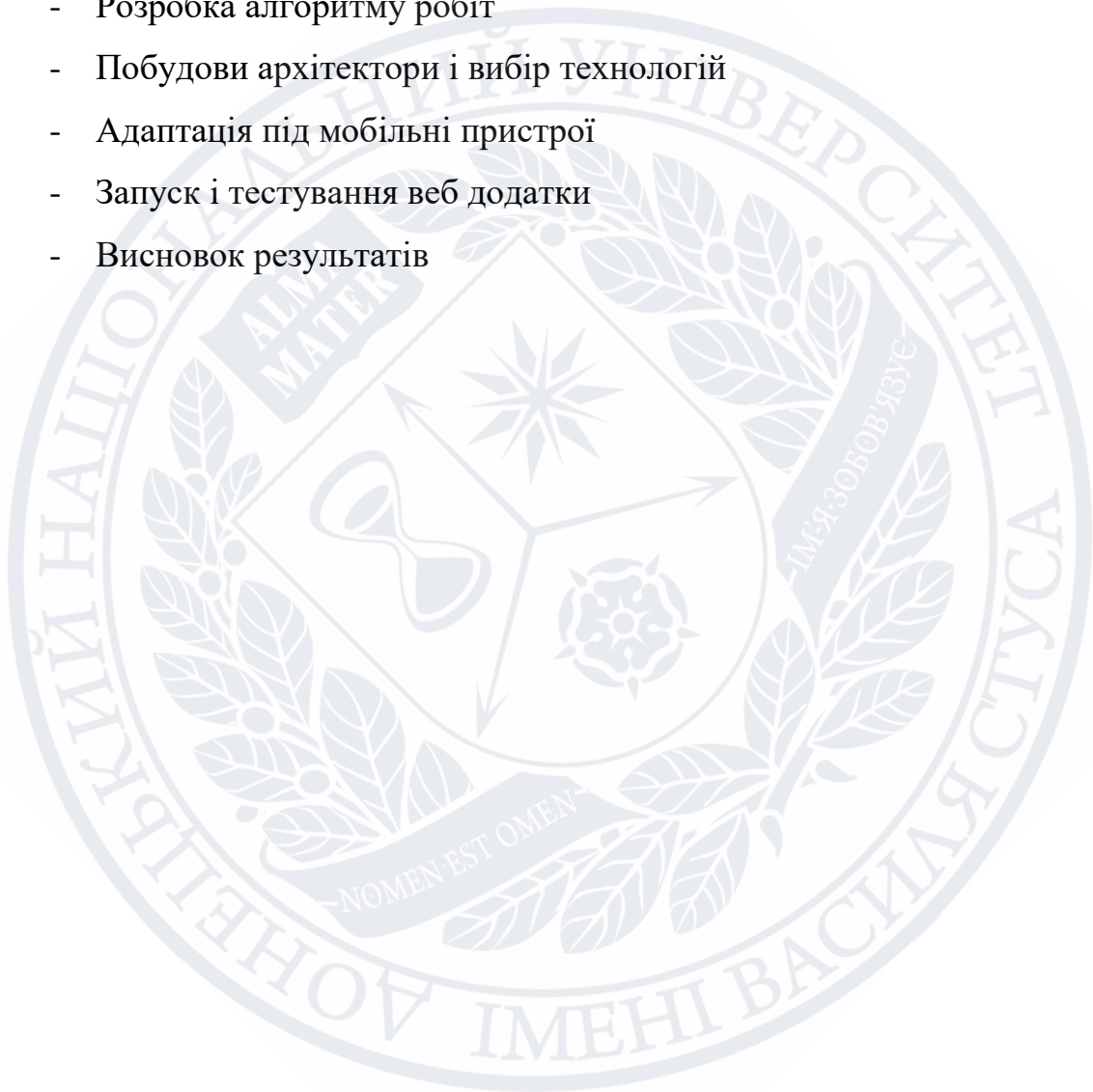
Ще важливий фактор це колірна гамма, вона повинна приємно виглядати в очах, і щоб приємно було взаємодіяти з веб додатком. Ми знаємо що різний набір відтінків викликає у людини різні почуття але ми будемо зосереджуватися на м'яких кольорах таких як зелений, рожевий і білий.

Побудуємо чітку розмітку елементів на клієнтському девайсу, щоб всі елементи рівно і чітко були розташовані. І зробимо адаптацію під різні пристрої щоб контент був однаково був зручний під усіма пристроями, для цього будемо

використовувати технологію flexbox яка дозволить уникнути статичних розмірів елементів і блоків і робить їх динамічними, щоб незалежно від девайсу блоки з елементами розтягувалися як гума, і при створенні контенту щоб були за них спокійні.

Підіб'ємо підсумки постановки задачі

- Знаходження проблеми і рішення їх
- Розробка алгоритму робіт
- Побудови архітектури і вибір технологій
- Адаптація під мобільні пристрої
- Запуск і тестування веб додатки
- Висновок результатів



2 ОПИС ВИМОГ ДЛЯ СТВОРЕННЯ ВЕБ ДОДАТКУ

2.1 Функціональні вимоги візуального дизайну

А тепер поговоримо про візуальний дизайн, чому він дуже важливий і чому він все вирішує.

Візуальний дизайн часто відкидають як цукерку для очей. Насправді ми можемо використовувати чотири ключові принципи візуального дизайну для створення більш корисних інтерфейсів. Ці принципи - це контраст, повторення, вирівнювання та близькість.

У сфері користувацького досвіду візуальний дизайн іноді сприймається як трохи сторонній. Справді, деякі візуальні дизайнери, мабуть, відчувають тиск на ребрендинг себе на дизайнера "досвіду користування" для просування своєї кар'єри. За моїм досвідом роботи з клієнтами, візуальний дизайн часто порівнюють з оформленням стін вашої вітальні. Це робить все трохи приємніше, але навряд чи це ключова частина архітектури вашого будинку.

Насправді візуальний дизайн має важливий вплив на зручність використання дизайну.

Дослідження показують, що люди вважають, що більш привабливі дизайни простіші у використанні, ніж менш привабливі - навіть тоді, коли вони цього не роблять. У психології це відомо як ефект естетичної юзабіліті, а у фольклорі простіше - «перші враження мають значення».

Але хороший візуальний дизайн пропонує більше, ніж покращення ставлення людей до дизайну.

Хороший візуальний дизайн насправді полегшить користування інтерфейсами.

Існує принаймні чотири ключові принципи візуального дизайну, які мають важливий вплив на зручність використання. Ці чотири принципи -

Яскраві ідеї для дизайнерів з досвідом роботи! Фокус на користувача контраст, повторення, вирівнювання та близькість - спочатку привабливу аббревіатуру КПВБ дав Робін Вільямс (візуальний дизайнер). Ви можете

використати ці чотири принципи, щоб зробити інтерфейси користувача більш привабливими та простішими у використанні.

Давайте розглянемо кожен по черзі.

Контраст

Контраст у візуальному дизайні допомагає спрямувати погляд глядача на те, що важливо, і допомагає зосередитись на тому, що робити далі. Це означає, що ваш заклик до дії - або що завгодно фокус уваги - на сторінці - дуже відрізняється від інших предметів, які її оточують.

У цьому прикладі рисунок 2.1 кнопка “Додати в кошик” відформатована ідентично кнопці “Скасувати”. Це означає, що користувачеві доведеться уважно прочитати текст на кожній кнопці перед тим, як зробити вибір.



Рисунок 2.1 - Дві кнопки на формі, одна з написом "Додати до кошика", а інша - "Скасувати". Оскільки дві кнопки відформатовані однаково, не зрозуміло, що є основною, а яка вторинною дією.

За умови хорошого використання контрасту ми можемо зробити вибір за замовчуванням відразу очевидним для користувачів. Наприклад, ми можемо:

- Зменшіть видиму глибину кнопки.
- Використовуйте римський (на відміну від жирного) шрифт.
- Видаліть кнопку повністю та змініть її на гіперпосилання.

На рисунку 2.2 показано вплив внесення цих змін, і з кожною зміною стає більш зрозумілим, який вибір ми очікуємо від користувача.

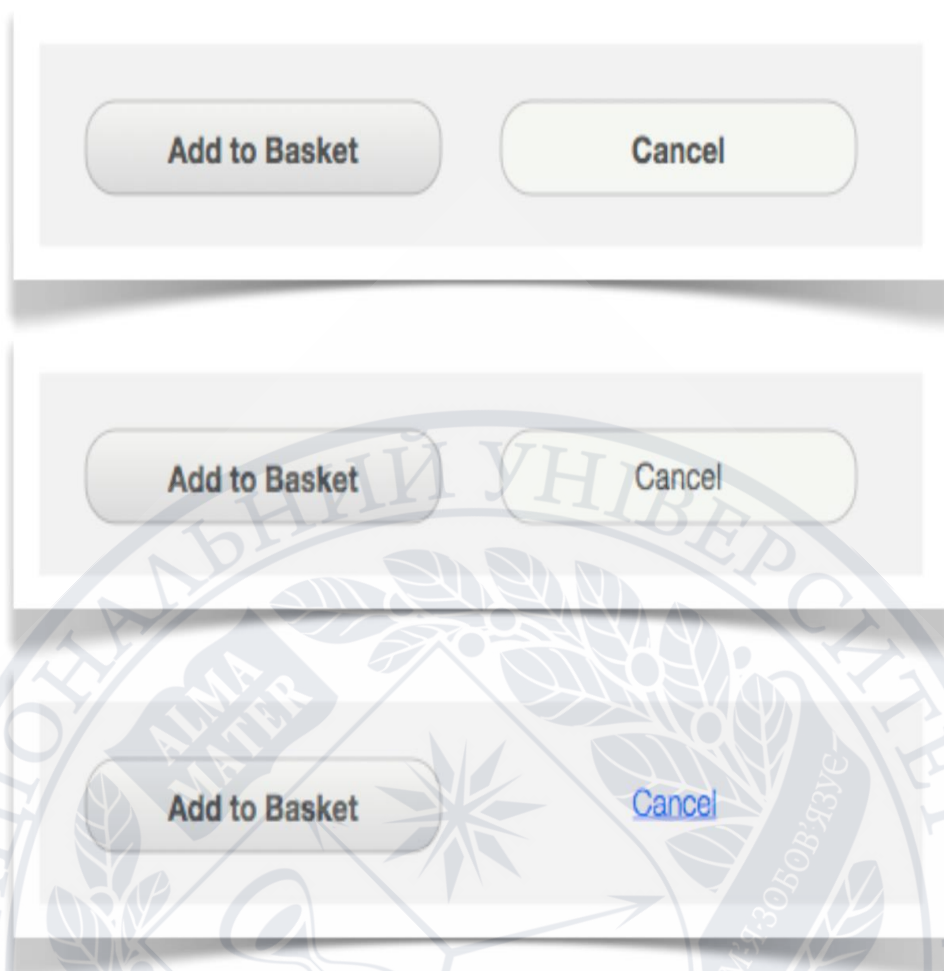


Рисунок 2.2 - Три приклади використання контрасту для відмежування первинної дії від вторинної. Перший приклад видалив видиму глибину кнопки “Скасувати”; другий приклад спирається на це і змінює шрифт; а в третьому прикладі кнопку замінено гіперпосиланням.

Цей приклад показує, що у вас є багато простих способів створити контраст у своєму інтерфейсі, навіть якщо ви вважаєте, що ваші навички дизайну обмежені. Прості зміни тексту, такі як напівжирний, курсив, підкреслення, великі літери, кольори та виділення, часто можуть бути достатніми (хоча, очевидно, їх потрібно використовувати в міру).

Повторення

Повторення насправді полягає у послідовності. (На жаль, „Послідовність“ не робить запам’ятовуваного аббревіатури в поєднанні з іншими принципами, тому замість цього ми виконуємо „Повторення“). На одному рівні цей принцип

просто стверджує, що системи є більш придатними для використання (і їх легше засвоювати), коли подібні речі подаються подібними способами.

Зверніть увагу, що це означає як внутрішню узгодженість (у межах вашої програми), так і зовнішню узгодженість (з платформою, для якої ви розробляєте). Для веб-сайту внутрішня узгодженість означає переконання, що ви використовуєте однакові шрифти, піктограми, заголовки, посилання, стилі списків та макет сторінки. Зовнішня узгодженість означає такі речі, як використання стандартних кнопок, кольорів посилань та результатів пошуку. Як об Нільсен навіть створив для цього "Закон": "Користувачі проводять більшу частину часу на інших веб-сайтах". Очікування людей формуються тим, що вони відчують на інших веб-сайтах, тому, якщо ви зробите це по-іншому, вашим веб-сайтом буде важче користуватися. Тож переконайтеся, що люди можуть знайти загальні елементи веб-сторінки (такі як заголовки сторінок, навігація сайтом, навігація сторінками, пошук тощо) у стандартних місцях.

Одним з найкращих способів розумно використовувати цей принцип - це продумати завдання, яке виконують ваші користувачі. Розглянемо такий приклад: якби я попросив вас прибрати ножі, виделки та ложки на моїй кухні, ви б шукали піднос для столових приборів. Але якби я тоді попросив вас прибрати канцелярський ніж, ви б, напевно, шукали переповнену шухляду, в якій є такі предмети, як штопор та солодки. І якби я тоді попросив вас прибрати коробку, ви, мабуть, поставили б її в гаражі за допомогою молотка, викруток та інших інструментів. Це здається суперечливим: усі вони ножі, то навіщо їх класти в різні місця?

Існує глибша причина, чому послідовність має значення в дизайні інтерфейсу користувача, і цей приклад це ілюструє. Ваша поведінка в цих прикладах повністю відповідає завданням, для якого ви використовуєте ці об'єкти. Ніж більше нагадує виделку, ніж коробковий ніж, коли завдання їсть; але це більше схоже на молоток, коли завдання - "зроби сам". Завдання - спільний знаменник.

Тож, думаючи про те, як застосувати повторення у своєму інтерфейсі користувача, подумайте про завдання, які виконують користувачі, і про те, як ви можете підтримати це своїм візуальним дизайном.

Вирівнювання

Вирівнювання просто означає переконатися, що всі елементи дизайну вирівнюються по горизонталі та вертикалі. Цього найкраще досягти, спроектувавши інтерфейс до базової сітки.

Вирівнювання - це, мабуть, найдраматичніша візуальна обробка дизайну, щоб зробити його візуально простішим у використанні. Погляньте на Рисунок 2.3 нижче. Не надто замислюючись над цим, яку форму виглядає легше заповнити, ту, що зліва чи ту, що справа?

The image shows two versions of a travel form side-by-side. Both forms are titled 'Traveler #1 - Adult' and include a note: 'Notes: Spelling of names must exactly match your government issued ID'. The forms contain the following fields: Title (dropdown), First Name (text), Last Name (text), Gender (dropdown), Date of Birth (DD/MM/YYYY), Email address (text), Phone number (text, with a note '(Ext: 800 622 3030)'), Password (text, with a note '(min: 8, max: 16, must contain 1 uppercase, 1 lowercase, 1 number, 1 special character)'), and TSA Secure Flight Program Information (checkbox). The TSA section includes fields for Optional TSA Items, Passport Number, Expiry Date (DD/MM/YYYY), Passport Issuing Country (dropdown), and Known Traveler Number (text). The left version of the form has a more complex layout with more vertical lines, while the right version is simpler and more horizontally aligned.

Рисунок 2.3 - Дві однакові форми, які відрізняються лише використанням вирівнювання.

Більшість людей кажуть, що форму праворуч виглядає простіше заповнити. Ці дві форми однакові, за винятком того, що поля форми були правильно вирівняні у прикладі праворуч. На Рисунку 1.4 я наклав кожен форму червоними лініями, щоб показати різні лінії вирівнювання. Ви бачите, що форма зліва, яку більшість людей описує як складнішу, також має більше вертикальних ліній вирівнювання.

Рисунок 2.4 - Це показує ті самі форми, що і на малюнку 2.3, але тепер червоні лінії показують точки вертикального вирівнювання різних елементів інтерфейсу користувача. Форма зліва (яку більшість людей описує як складнішу) також має більше ліній вертикального вирівнювання.

Близькість

Принцип близькості був вперше сформульований гештальт-школою психології. Теоретики Гештальта наголошували, що візуальне сприйняття стосується сприйняття організованих цілих, а не лише бачення ізольованих предметів. Принцип близькості означає, що якщо ви розміщуєте елементи в інтерфейсі користувача поруч один з одним, люди подумают, що вони якимось чином пов'язані. Існує два способи використання цього принципу для підвищення зручності використання. По-перше, близькість допоможе користувачам знайти варіант, який вони шукають. Якщо користувачі хочуть побачити ваш асортимент ноутбуків, їм буде швидше це знайти, якщо він згрупований з іншими супутніми елементами в частині інтерфейсу, присвяченій «Комп'ютери та офіс», ніж якщо це один із багатьох варіантів у неорганізованому безлад на сторінці дивиться рисунок 2.5

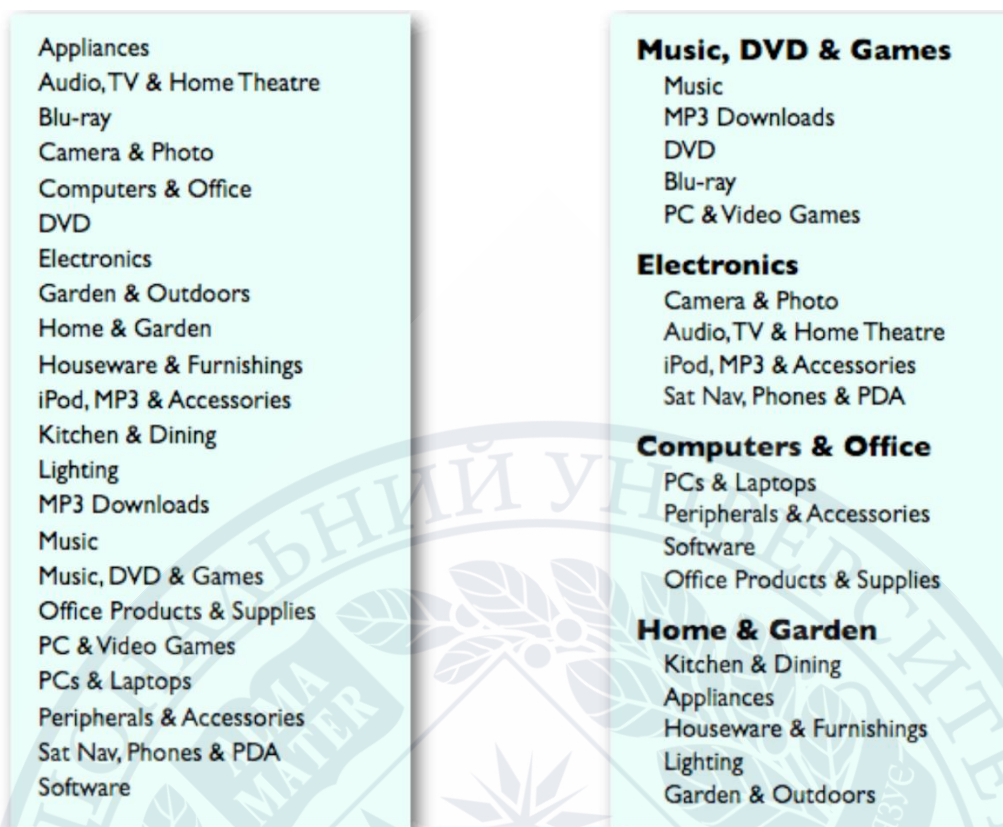


Рисунок 2.5 - Простий приклад, що демонструє принцип близькості, що використовується для впорядкування елементів у навігаційному меню. Елементи ліворуч розташовані в алфавітному порядку, тоді як ті, що розташовані праворуч, упорядковані в групи з заголовком кожної групи.

Ви можете застосувати цей принцип в інтерфейсі користувача за допомогою відповідного використання кольорів тла, меж та пробілів: це допоможе користувачам ідентифікувати набір елементів як дискретний функціональний блок.

Але є друга причина, чому цей принцип робить користувацькі інтерфейси більш корисними: спосіб упорядкування інформації на сторінці допомагає людям побудувати концептуальну модель структури інтерфейсу. Тенденція веб-дизайну використання „портлетів” (як на домашній сторінці BBC) є гарним прикладом цього.

Проекти КПВБ - це більш придатні конструкції

Спокусливо відкинути візуальний дизайн як просту глазур чи цукерку для очей. І звичайно, ви ніколи не врятуєте жахливий інтерфейс, просто зробивши його більш привабливим. ("Можна нанести помаду на свиню", як каже стара

приказка, "але це все-таки свиня"). Але неправильно думати про хороший візуальний дизайн як про щось, що спонукає до першого використання. Дотримання принципів проектування КПВБ насправді робить системи більш придатними для використання.



2.2 Метод розробці

Для нашої розробці треба вибрати ключовий підхід. Як команди Scrum працюють у ітераціях часу, зосереджуючись на невеликих завданнях, створених із більших завдань, встановлених на обмеженні WIP для всього Спринту (ітерації). Для порівняння, Канбан надає велике значення візуалізації робочого процесу та обмеженню незавершеного виробництва протягом певного часу (кількість завдань на одного члена команди або на кожну команду).

Загальні відмінності. Стосовно цих методів по відношенню один до одного, Книберг і Скарін підкреслюють той факт, що їх не обов'язково порівнювати між собою, оскільки вони часто виконують різні види роботи. Але загалом відомо, що Scrum накладає набагато більше обмежень на робочий процес, тоді як Kanban надає гнучкість і зосереджується на можливості вдосконалення. Scrum призначає ролі, вимагає ітерацій із обмеженим часом, Kanban підтримує постійний потік завдань. Обидві методології будують майбутні вдосконалення на аналізі минулого досвіду.

Scrum закритий, поки Канбан залишається відкритим для змін. Основна відмінність - опір Скрума змінам під час ітерації та готовність Канбана до змін у будь-який час. Іншою явною відмінністю є той факт, що плата Scrum скидається після кожної ітерації, і Kanban може продовжувати, оскільки відставання постійно поповнюється. Крім того, Scrum просить нашу команду розділити на багатофункціональні менші групи та вимагає кошторисів завершення проекту - жоден з яких не потрібен у Канбані.

Загальна подібність. Що стосується схожості Scrum - Kanban: вони обидва є Lean і Agile, і обидва сприяють обробці кількох проектів одночасно. Автори книги закликають нас не обмежуватися лише однією методологією, припускаючи, що Скрумбан може бути шляхом для багатьох команд. Вони також надають кілька прикладів реальних дощок Agile та дають корисні поради щодо того, як розпочати Agile.

Висновок. Їх остаточні пункти на винос - це ніколи не зупиняти ретроспекцію, щоб вчитися на своєму досвіді та продовжувати експериментувати, оскільки ви ніколи не знаєте, яке можливе рішення може виявитися найкращим для вас.



3 ШАБЛОНИ ПРОЕКТУВАННЯ ДОДАТКУ

Що таке шаблоне проектування. Шаблон проектування - це багаторазове рішення, яке можна застосувати до найпоширеніших проблем у розробці програмного забезпечення - в нашому випадку - при написанні веб-додатків JavaScript. Інший спосіб розглядати шаблони - це шаблони того, як ми вирішуємо проблеми - такі, які можна використовувати у багатьох ситуаціях.

Отже, чому важливо розуміти закономірності та бути з ними знайомими? Шаблони проектування мають три основні переваги:

1. Шаблони - це перевірені рішення: вони забезпечують надійні підходи до вирішення проблем у розробці програмного забезпечення із використанням перевірених методів, що відображають досвід та розуміння розробників, які допомогли визначити їх, щоб привести їх до моделі.
2. Шаблони можна легко використовувати повторно: візерунок, як правило, відображає нестандартне рішення, яке можна адаптувати відповідно до наших власних потреб. Ця особливість робить їх досить надійними.
3. Шаблони можуть бути виразними: коли ми розглядаємо шаблон, як правило, існує набір структури та словниковий запас для представленого рішення, які можуть допомогти досить елегантно висловити досить великі рішення.

Шаблони не є точним рішенням. Важливо, щоб ми пам'ятали, що роль шаблону полягає лише у наданні нам схеми рішення. Шаблони не вирішують усіх дизайнерських проблем і не замінюють хороших дизайнерів програмного забезпечення, проте підтримують їх. Далі ми розглянемо деякі інші переваги, які можуть запропонувати моделі.

- Повторне використання шаблонів допомагає запобігти незначним проблемам, які можуть спричинити серйозні проблеми в процесі розробки додатків. Що це означає, коли код побудований на перевірених зразках, ми можемо дозволити собі витрачати менше часу, турбуючись про структуру нашого коду, і більше часу, фокусуючись на якості нашого загального

рішення. Це пояснюється тим, що шаблони можуть заохотити нас до більш структурованого та організованого кодування, уникаючи необхідності переробляти його з метою чистоти в майбутньому.

- Шаблони можуть надати узагальнені рішення, які задокументовані таким чином, що не вимагають прив'язки їх до певної проблеми. Цей узагальнений підхід означає, що незалежно від програми (і в багатьох випадках мови програмування), з якою ми працюємо, шаблони проектування можуть застосовуватися для поліпшення структури нашого коду.
- Деякі шаблони можуть насправді зменшити загальний розмір файлу нашого коду, уникаючи повторення. Заохочуючи розробників уважніше розглядати свої рішення для областей, де можна негайно зменшити кількість повторень, наприклад зменшуючи кількість функцій, що виконують подібні процеси, на користь однієї узагальненої функції, загальний розмір нашої кодової бази може бути зменшений. Це також відомо як зробити код більш сухим.
- Шаблони додають до словника розробника, що робить спілкування швидшим.
- Часто використовувані шаблони з часом можна вдосконалити, використовуючи колективний досвід, який інші розробники, використовуючи ці шаблони, вносять назад у спільноту дизайнерських шаблонів. В одних випадках це призводить до створення цілком нових зразків дизайну, тоді як в інших це може призвести до надання вдосконалених вказівок щодо того, як найкраще використовувати конкретні зразки. Це може забезпечити, щоб рішення на основі шаблонів продовжували ставати більш надійними, ніж можуть бути спеціальні рішення.

Щоб зрозуміти, наскільки корисними можуть бути шаблони, давайте розглянемо дуже просту задачу вибору елементів, яку вирішує нам бібліотека jQuery.

Уявіть, що у нас є сценарій, де для кожного елемента DOM, знайденого на сторінці з класом "foo", ми хочемо збільшити лічильник. Який найефективніший спосіб запитувати цю колекцію елементів? Ну, є кілька різних способів вирішення цієї проблеми:

1. Виділіть усі елементи на сторінці, а потім збережіть посилання на них. Далі відфільтруйте цю колекцію та використовуйте регулярні вирази (або інші засоби), щоб зберігати лише ті, що мають клас "foo".
2. Використовуйте сучасну функцію власного браузера, наприклад `querySelectorAll()`, щоб виділити всі елементи з класом "foo".
3. Використовуйте власну функцію, таку як `getElementsByClassName()`, щоб таким же чином повернути бажану колекцію.

Отже, який із цих варіантів є найшвидшим? Це насправді варіант 3. у 8-10 разів більший за альтернативи. Однак у реальному додатку 3. не працюватиме у версіях Internet Explorer нижче 9, тому необхідно використовувати 1., де і 2., і 3. не підтримуються.

Однак розробникам, які використовують jQuery, не потрібно турбуватися про цю проблему, оскільки вона, на щастя, абстрагована для нас за допомогою шаблону Фасад. Як ми розглянемо більш докладно пізніше, цей шаблон забезпечує простий набір абстрагованих інтерфейсів (наприклад, `$el.css()`, `$el.animate()`) для декількох більш складних основних тіл коду. Як ми вже бачили, це означає, що менше часу потрібно турбуватися про деталі рівня реалізації.

За лаштунками бібліотека просто обирає найбільш оптимальний підхід до вибору елементів залежно від того, що підтримує наш поточний браузер, і ми просто споживаємо шар абстракції.

Ми, мабуть, також знайомі з \$ ("селектор") jQuery. Це набагато простіше у використанні для вибору елементів HTML на сторінці порівняно з необхідністю вибирати вручну `getElementById()`, `getElementsByClassName()`, `getElementsTagName()` тощо.

Хоча ми знаємо, що `querySelectorAll()` намагається вирішити цю проблему, порівняйте зусилля, які стосуються використання інтерфейсів фасаду jQuery, та вибір самих оптимальних шляхів вибору. Конкурсу немає! Абстракції з використанням шаблонів можуть запропонувати реальну цінність

Тестування шаблонів, про-шаблон та правило трьох

Пам'ятаєм, що не кожен алгоритм, найкраща практика чи рішення представляє те, що можна вважати повноцінним зразком. Тут може бути кілька ключових інгредієнтів, яких немає, і спільнота зразків, як правило, насторожено ставиться до чогось, що претендує на те, що воно є, якщо це не було суто перевірено. Навіть якщо нам представлено щось, що, як видається, відповідає критеріям шаблону, це не слід вважати таким, поки воно не пройде відповідних періодів перевірки та тестування іншими.

Вивчаючи шаблони проектування, нерегулярно зустріти термін "прото-шаблон". Що це? Ну, шаблон, про який ще не було відомо, що пройшов тести на "шаблон", зазвичай називають прото-шаблоном. Прото-зразки можуть виникати внаслідок роботи когось, хто встановив конкретне рішення, яке варто донести до спільноти, але, можливо, ще не мало можливості бути суто перевіреним через дуже молодий вік.

В якості альтернативи, особа (особи), що ділиться зразком, може не мати часу чи інтересу, щоб пройти процес „зразка”, і замість цього може опублікувати короткий опис свого прото-зразка. Короткі описи або фрагменти цього типу зразків відомі як патлети.

Робота з повного документування кваліфікованого зразка може бути досить лякаючою. Оглядаючись на деякі найдавніші роботи в галузі дизайнерських зразків, візерунок можна вважати «хорошим», якщо він робить наступне:

- Вирішує конкретну проблему: Шаблони не повинні просто фіксувати принципи або стратегії. Їм потрібно захоплювати рішення. Це один з найважливіших інгредієнтів для гарного малюнка.
- Рішення цієї проблеми не може бути очевидним: ми можемо виявити, що методи вирішення проблем часто намагаються вивести з відомих перших принципів. Найкращі шаблони проектування зазвичай пропонують вирішення проблем опосередковано - це вважається необхідним підходом до найскладніших проблем, пов'язаних з дизайном.
- Описана концепція повинна бути доведеною: зразки дизайну вимагають підтвердження того, що вони функціонують, як описано, і без цього доказу конструкцію не можна серйозно розглянути. Якщо модель має дуже спекулятивний характер, лише сміливі можуть намагатися використовувати її.
- Він повинен описувати взаємозв'язок: У деяких випадках може здатися, що шаблон описує тип модуля. Хоча реалізація може виглядати таким чином, офіційний опис шаблону повинен описувати набагато глибші системні структури та механізми, що пояснюють її зв'язок із кодом.

Нам було б пробачено думати, що прото-шаблон, який не відповідає настановам, не вартий того, щоб навчитися, проте, це далеко не правда. Багато прото-шаблонів насправді цілком непогані. Я не кажу, що всі прото-зразки варті того, щоб їх розглянути, але в природі є чимало корисних, які можуть допомогти нам у розробці веб додатку.

Структура шаблонного проектування

Шаблону може підійти до структури структури, реалізації та призначення нового шаблону. Спочатку шаблон подається у вигляді правила, яке встановлює взаємозв'язок між:

- Контекст
- Система сил, що виникає в цьому контексті і
- Конфігурація, яка дозволяє цим силам вирішити себе в контексті

Маючи це на увазі, давайте зараз подивимось короткий опис елементів компонента для шаблону дизайну. Шаблон дизайну повинен мати:

- Назва шаблону та опис
- Контекстна схема - контексти, в яких шаблон ефективно реагує на потреби користувачів.
- Постановка проблеми - постановка проблеми, яка вирішується, щоб ми могли зрозуміти намір шаблону.
- Рішення - опис того, як вирішується проблема користувача, у зрозумілому переліку кроків та уявлень.
- Дизайн - опис дизайну шаблону, зокрема поведінки користувача у взаємодії з ним
- Впровадження - посібник з того, як буде реалізований шаблон
- Ілюстрації - візуальне зображення класів у зразку (наприклад, схема)
- Приклади
- Реквізити
- Відносини
- Відоме використання
- Дискусії

Написання шаблонів проектування

Написання хороших зразків є складним завданням. Шаблони не тільки повинні (в ідеалі) забезпечувати значну кількість довідкового матеріалу для кінцевих споживачів, але вони також повинні мати можливість захищати, чому вони необхідні.

Не завжди зрозуміло, чи шматок коду, який ми розглядаємо, дотримується встановленого шаблону або просто випадково виявляється таким, яким він є. Коли ми розглядаємо корпус коду, який, на нашу думку, може використовувати шаблон, нам слід задуматися про те, щоб записати деякі аспекти коду, які, на нашу думку, підпадають під певний існуючий шаблон або набір шаблонів.

У багатьох випадках аналізу шаблонів ми можемо виявити, що ми просто розглядаємо код, який відповідає добрим принципам та практиці дизайну, які можуть випадково збігатися з правилами для шаблону. Но рішення, у яких не з'являються ні взаємодії, ні визначені правила, не є шаблонами.

Дослідуючи структуру та семантику -ми робимо взаємодію та контекст моделей, які нас цікавлять, щоб ми могли визначити принципи, які допомагають організувати ці шаблони разом у корисних конфігураціях.

Після того, як ми ознайомимося з великою кількістю інформації про літературу з шаблонів, ми, можливо, захочемо почати писати наш шаблон, використовуючи існуючий формат, і подивитися, чи зможемо ми подумати про нові ідеї щодо його вдосконалення або інтеграції своїх ідей.

Для розробки нового шаблонного проектування ми будемо дотриматися:

- Практичності
- Найкраща практика
- Шаблони дизайну повинні бути повністю прозорими для будь-якого типу користувацького досвіду
- Оригінальність не є ключовою у дизайні зразків: під час написання зразка нам не потрібно бути першовідкривачем рішень, що документуються, і нам не доведеться турбуватися про те, що наш дизайн перекривається незначними шматками інших зразків. Якщо підхід досить сильний, щоб мати широку корисну застосовність, він має шанс бути визнаним дійсним зразком.
- Шаблони потребують вагомому набору прикладів: За хорошим описом зразків повинен слідувати не менш сильний набір прикладів, що демонструють успішне застосування нашого зразка. Щоб показати широке використання, ідеально підходять приклади, що демонструють хороші принципи дизайну.

Написання зразків - це ретельний баланс між створенням загального, конкретного та, перш за все, корисного дизайну. Спробуйте переконатися, що якщо ви пишете візерунок, ви охоплюєте найширші сфери застосування, і ви

повинні бути добре. Я сподіваюся, що цей короткий вступ до зразків написання дав вам деякі ідеї, які допоможуть вам у процесі навчання для наступних розділів цієї книги.

Протилежний вид шаблонного проектування

Якщо ми вважаємо, що шаблон являє собою найкращу практику, анти-шаблон - це урок, який було засвоєно. Термін "анти-візерунки" був введений в 1995 році Ендрю Кенігом у листопадовому звіті C++ того року, натхненний книгою GoF "Шаблони дизайну". У звіті Кеніга представлено два поняття анти-шаблонів.

Анти-візерунки:

- Опишіть погане вирішення певної проблеми, яка призвела до поганої ситуації
- Опишіть, як вийти із згаданої ситуації та як від цього знайти гарне рішення

На цю тему Олександр пише про труднощі в досягненні гарного балансу між хорошою структурою дизайну та хорошим контекстом:

«Ці примітки стосуються процесу проектування; процес винаходу фізичних речей, які відображають новий фізичний порядок, організацію, форму у відповідь на функцію ... кожна проблема проектування починається із зусиль для досягнення придатності між двома сутностями: формою, про яку йдеться, та її контекстом. Форма - це рішення проблеми; контекст визначає проблему».

Незважаючи на те, що досить важливо знати про шаблони дизайну, не менш важливо розуміти і анти-шаблони. Давайте визначимо причину цього. При створенні програми життєвий цикл проекту починається з побудови, однак після того, як ви зробили початковий випуск, його потрібно підтримувати. Якість остаточного рішення буде або хорошою, або поганою, залежно від рівня майстерності та часу, який команда вклала в нього. Тут добре і погане розглядаються в контексті - „ідеальний” дизайн може кваліфікуватися як анти-шаблон, якщо застосовуватись у неправильному контексті.

Більші проблеми виникають після того, як програма потрапила у виробництво і готова перейти в режим обслуговування. Розробник, який працює над такою системою і раніше не працював над додатком, може випадково внести в проект

поганий дизайн. Якщо згадані погані практики створюються як анти-шаблони, вони дозволяють розробникам заздалегідь розпізнати їх, щоб вони могли уникнути типових помилок, які можуть виникнути - це паралельно способу, в якому шаблони дизайну надають нам спосіб розпізнати загальні техніки, які є корисними.

Підводячи підсумок, анти-шаблон - це поганий дизайн, який гідний документування. Прикладами анти-шаблонів у JavaScript є наступні:

- Забруднення глобального простору імен шляхом визначення великої кількості змінних у глобальному контексті
- Передача рядків, а не функцій, або `setTimeout`, або `setInterval`, оскільки це ініціює використання `eval()` внутрішньо.
- Модифікація прототипу класу `Object` (це особливо поганий шаблон)
- Використання JavaScript у вбудованій формі, оскільки це негнучко
- Використання `document.write` там, де доречніші альтернативи DOM, такі як `document.createElement`. Документ.write був грубо зловживаний протягом багатьох років і має чимало недоліків, включаючи те, що якщо він виконується після завантаження сторінки, він може фактично перезаписати сторінку, на якій ми перебуваємо, тоді як `document.createElement` - ні. Ми можемо побачити тут живий приклад цього в дії. Він також не працює з XHTML, що є ще однією причиною, що обирає більш зручні для DOM методи, такі як `document.createElement`.

Знання анти-шаблонів має вирішальне значення для успіху. Як тільки ми зможемо розпізнати такі анти-шаблони, ми зможемо рефакторингувати наш код, щоб заперечити їх, щоб загальна якість наших рішень миттєво покращилася.

Категорії шаблонного проектування

У глосарії з відомої книги дизайну «Домен-керовані умови» справедливо зазначається, що:

«Шаблон дизайну називає, абстрагує та визначає ключові аспекти загальної структури дизайну, що робить його корисним для створення об'єктно-орієнтованого дизайну, що використовується багаторазово. Шаблон дизайну

визначає класи, що беруть участь, та їх екземпляри, їх ролі та співпрацю, а також розподіл обов'язків

Кожен шаблон дизайну фокусується на конкретній об'єктно-орієнтованій проблемі дизайну або проблемі. Він описує, коли він застосовується, чи можна застосовувати його з огляду на інші конструктивні обмеження, а також наслідки та компроміси від його використання. Оскільки ми повинні врешті-решт реалізувати наші проекти, шаблон дизайну також надає зразок ... коду для ілюстрації реалізації.

Хоча шаблони проектування описують об'єктно-орієнтовані проекти, вони базуються на практичних рішеннях, які були впроваджені в основних об'єктно-орієнтованих мовах програмування "

Шаблони дизайну можна розділити на безліч різних категорій. У цьому розділі ми розглянемо три з цих категорій і коротко згадаємо кілька прикладів зразків, які підпадають під ці категорії, перш ніж детальніше досліджувати конкретні.

Творчі шаблони дизайну. Шаблони креативного дизайну зосереджуються на обробці механізмів створення об'єктів, коли об'єкти створюються таким чином, що відповідає ситуації, в якій ми працюємо. Базовий підхід до створення об'єктів в іншому випадку може призвести до додаткової складності в проекті, тоді як ці шаблони мають на меті вирішити цю проблему шляхом контроль процесу створення.

Деякі з візерунків, які підпадають під цю категорію: конструктор, фабрика, реферат, прототип, синглтон і будівельник

Структурні шаблони дизайну. Структурні структури стосуються композиції об'єктів і, як правило, визначають прості способи реалізації взаємозв'язків між різними об'єктами. Вони допомагають гарантувати, що коли одна частина системи змінюється, вся структура системи не повинна робити те саме. Вони також допомагають у переробці частин системи, які не відповідають певній цілі, в ті, що відповідають.

Шаблони, які підпадають під цю категорію, включають: декоратор, фасад, напівважкий вага, адаптер та проксі.

Поведінкові шаблони дизайну

Поведінкові моделі зосереджені на поліпшенні або впорядкуванні зв'язку між різними об'єктами в системі.

Деякі моделі поведінки включають: Ітератор, Посередник, Спостерігач та Відвідувач.

Конструктор шаблон

У класичних об'єктно-орієнтованих мовах програмування конструктор - це спеціальний метод, що використовується для ініціалізації новоствореного об'єкта після того, як для нього виділено пам'ять. У JavaScript, оскільки майже все є об'єктом, нас найчастіше цікавлять конструктори об'єктів.

Конструктори об'єктів використовуються для створення конкретних типів об'єктів - як підготовки об'єкта до використання, так і прийняття аргументів, які конструктор може використовувати для встановлення значень властивостей-членів та методів при першому створенні об'єкта.

Модульний шаблон. Модулі є невід'ємною частиною архітектури будь-якої надійної програми і, як правило, допомагають підтримувати одиниці коду для проекту як чітко відокремленими, так і організованими.

Модульний шаблон

Модулі є невід'ємною частиною архітектури будь-якої надійної програми і, як правило, допомагають підтримувати одиниці коду для проекту як чітко відокремленими, так і організованими.

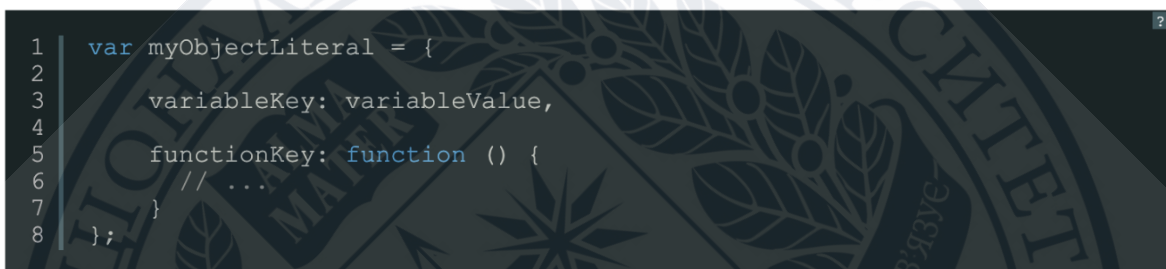
У JavaScript існує кілька варіантів реалізації модулів. До них належать:

- Шаблон модуля
- Буквене позначення об'єкта
- Модулі AMD
- Модулі CommonJS
- Модулі ECMAScript Harmony

Останні три з цих варіантів ми вивчимо пізніше у книзі в розділі Сучасні модульні шаблони дизайну JavaScript.

Шаблон модуля частково заснований на літералах об'єктів, тому має сенс спочатку оновити наші знання про них.

Об'єктні літерали. У буквальних позначеннях об'єкта об'єкт описується як сукупність пар імен / значень, розділених комами, укладених у фігурні дужки ({}). Імена всередині об'єкта можуть бути або рядками, або ідентифікаторами, після яких ставиться двокрапка. Після кінцевої пари ім'я / значення в об'єкті не повинно використовуватися кома, оскільки це може призвести до помилок.



```

1  var myObjectLiteral = {
2
3      variableKey: variableValue,
4
5      functionKey: function () {
6          // ...
7      }
8  };

```

Рисунок 3.1 – Методі в функції

Об'єктивні літерали не вимагають створення екземпляра за допомогою оператора new, але їх не слід використовувати на початку оператора, оскільки відкриття { може інтерпретуватися як початок блоку. Поза об'єктом нові члени можуть бути додані до нього за допомогою присвоєння наступним чином `myModule.property = "someValue";`

Нижче ми можемо побачити більш повний приклад модуля, визначеного за допомогою нотації об'єктового літералу:

```

var myModule = {

    myProperty: "someValue",

    // об'єкти-літерали можуть містити властивості та методи.
    // Наприклад, ми можемо визначити подальший об'єкт для конфігурації модуля:
    myConfig: {
        useCaching: true,
        language: "en"
    },

```

```
// дуже базовий метод
saySomething: function () {
  console.log( "Where in the world is Paul Irish today?" );
},

// вивести значення на основі поточної конфігурації
reportMyConfig: function () {
  console.log( "Caching is: " + ( this.myConfig.useCaching ? "enabled" : "disabled" )
);
},

// замінити поточну конфігурацію
updateMyConfig: function( newConfig ) {

  if ( typeof newConfig === "object" ) {
    this.myConfig = newConfig;
    console.log( this.myConfig.language );
  }
};

myModule.saySomething();

// Виходи: Кешування: увімкнено
myModule.reportMyConfig();

// Виходи: fr
myModule.updateMyConfig({
  language: "fr",
  useCaching: false
});

// Результати: Кешування: вимкнено
myModule.reportMyConfig();
```

Використання об'єктних літералів може допомогти в інкапсуляції та впорядкуванні коду, і Ребекка Мерфі раніше вже детально писала про цю тему, якщо ви хочете прочитати в об'єктних літералах далі.

Тим не менш, якщо ми обираємо цю техніку, ми можемо бути однаково зацікавлені в шаблоні модуля. Він як і раніше використовує літерали об'єктів, але лише як повернене значення з функції масштабування.

Модульний шаблон. Шаблон модуля спочатку визначався як спосіб забезпечити як приватне, так і публічне інкапсулювання для класів із звичайної інженерії програмного забезпечення.

У JavaScript шаблон Module використовується для подальшого наслідування концепції класів таким чином, що ми можемо включати як загальнодоступні / приватні методи, так і змінні всередині одного об'єкта, тим самим захищаючи певні частини від глобальної області дії. Це призводить до зменшення ймовірності конфлікту імен наших функцій з іншими функціями, визначеними в додаткових сценаріях на сторінці.

Конфіденційність. Шаблон модуля інкапсулює "конфіденційність", стан та організацію, використовуючи закриття. Він забезпечує спосіб обгортання поєднання загальнодоступних та приватних методів та змінних, захищаючи частини від витоку в глобальну сферу та випадкового зіткнення з інтерфейсом іншого розробника. За цим шаблоном повертається лише загальнодоступний API, зберігаючи все інше в рамках закриття приватним.

Це дає нам чисте рішення для екранування логіки, що робить важкі дії, одночасно виставляючи лише інтерфейс, який ми хочемо використовувати для інших частин нашого додатку. Шаблон використовує вираз функції, що викликається негайно (IIFE - докладніше про це див. У розділі про шаблони простору імен), де повертається об'єкт.

Слід зазначити, що насправді в JavaScript не існує явно справжнього відчуття "приватності", оскільки на відміну від деяких традиційних мов, він не має модифікаторів доступу. Технічно змінні не можна оголосити як загальнодоступними, так і приватними, тому ми використовуємо область функцій для імітації цієї концепції. У шаблоні модуля змінні або методи, які оголошені, доступні лише всередині самого модуля завдяки закриттю. Однак змінні або методи, визначені в об'єкті, що повертається, доступні кожному.

4 РОЗРОБКА ДОДАТКУ

4.1 Технології з яких будуємо додаток

Головна бібліотека для побудови веб додатки це React js, вона ж відповідає за інтерфейс веб додатки. React js був розробленої компанією Facebook, вони переосмислили модель розробки веб додатків і вирішили зробити так щоб веб сайт був як рідна програма, так само швидко переключалася і передавала дані на сервер і відображала нові дані.

React js задав нову планку розробки додатків, що багато програм і бізнес процесів перейшли в онлайн веб додатки. З стрімким розвитком веб сайтів все більше переходять в браузер що є думки що все буде тільки через браузер.

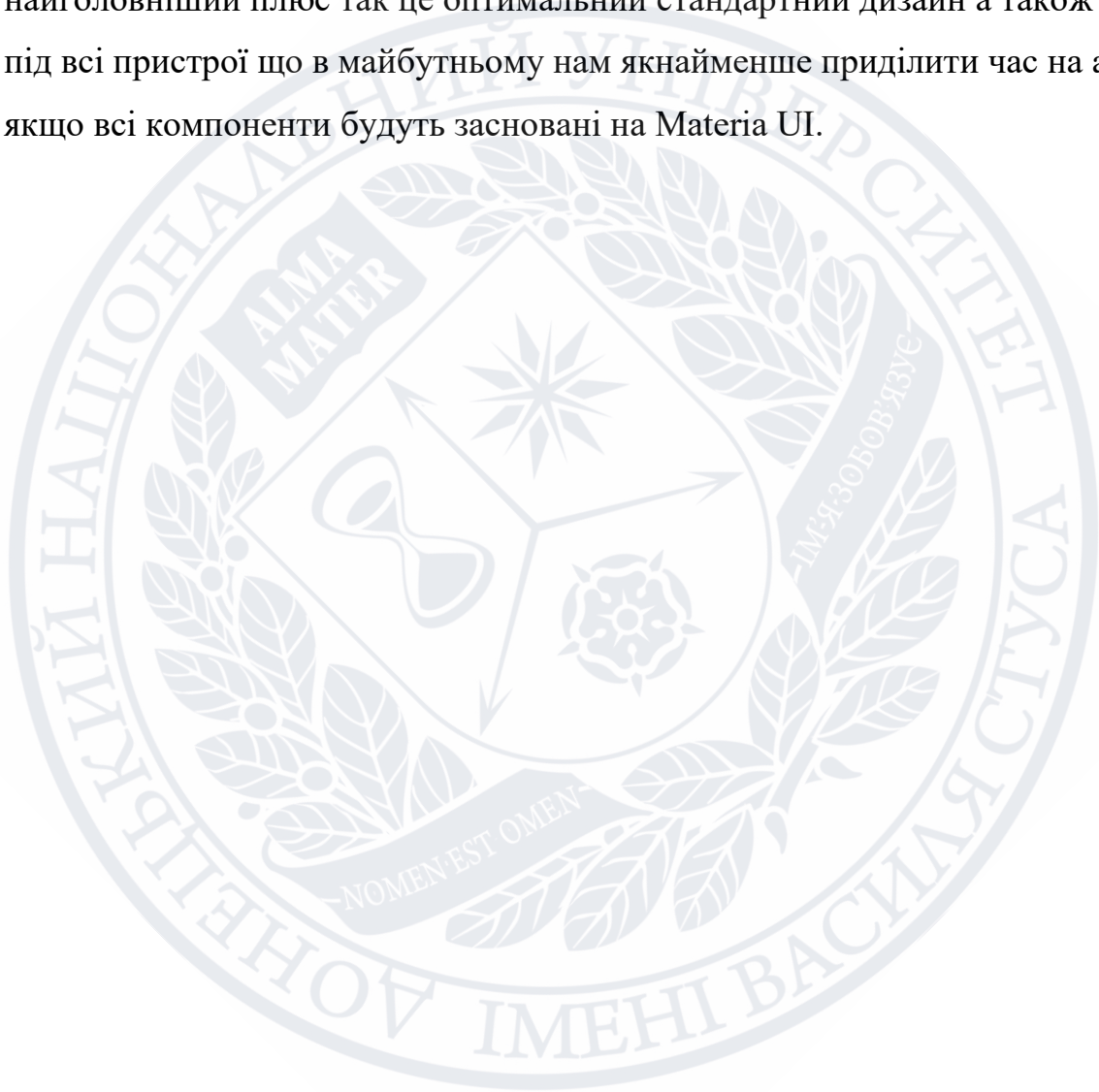
Бібліотека дозволяє розбивати проект на компоненти і на окремі модулі що дозволяє перевикористати, що прискорює розробку і дає легкість в підтримці, можна розробляти окремі компоненти окремо і впроваджувати в загальну картину тим самим ми підвищуємо якість продукту і масштабованість що в наші дні дуже важливо.

Також головним фактором у React js є це віртуальне дерево компонентів, воно допомагає оптимізувати додатки щоб швидко і чітко працював, якщо раніше до появи це бібліотеки дерево компонентів повністю пере оновлювався всього за одне компонента що тягнула все зміни програми та було сильно енерго затратно то з появою React js при зміні певного компонента інші компоненти не оновлюються просто так що в рази прискорює додатки.

Наступна бібліотека це Redux js, вона відповідає за стан додатки, і за контролювання його. Наприклад у нас є різні компоненти в різних частинах програми, але не пов'язані між собою але потрібно щоб вони працювали з однаковими даними та з однаковим станом що взаємопов'язувати ці компоненти, то на допомогу приходить Redux js. Як він працює, він обертає кожен з програм у сховище і створює для кожної сутності свій стан і стан суті має дії, кожне дії виконує різні потреби зі станом і щоб розуміти з яким дією працюємо ми

створюємо окремо дії і їм присвоюємо типи, і за типами викликає дію а дії тягне за собою зміни стану в додатку.

Для швидкої побудови компонентів і великих сутностей складається з множин компонентів нам будуть потрібні допоміжна інтерфейсна бібліотека і це буде Materia UI. Вона нам дає готовий фундамент бібліотеки і ми можемо її перевикористати в свої потреби, можемо переписати стилі та розміри, але найголовніший плюс так це оптимальний стандартний дизайн а також адаптація під всі пристрої що в майбутньому нам якнайменше приділити час на адаптацію якщо всі компоненти будуть засновані на Materia UI.



4.2 Архітектура додатку

Проектуючи архітектура потрібно завжди дотримуватися простоти і чистоти, щоб було зрозуміло з першого разу зрозуміти що де є і як з цим працювати, легкий спосіб це перевірити попросити людину не знайому з певною архітектурою щоб він зміг щось зробити чи масштабування або що то переробити. Адже з розширенням проекту все складніше підтримувати і встежити всі помилки і недоліки.

Ми почнемо з роутінга сторінок, ми створимо папку з роутингом і туди будуть імпортуватися всі сторінки і ми обгорнемо перевірки на кожен сторінку, де і для яких користувачів можна отримати доступ до певної сторінки.

Паралельно з цим ми зробимо папку зі сторінками де кожна сторінка буде відповідати за певний скрін і вони будуть тягнутися в роутинг, потім ми беремо роутинг тягнемо до головного компоненту обгортці де ми підкручуємо наш стейт менеджер і компоненти верхнього порядку, наприклад компонент тостер буде в найголовнішому компоненті наприклад, виклику його ми зможемо всюди побачити тостер у всіх сторінках що при використовуємо один код всюди.

Зробимо папку компонентів де у нас будуть порожні компоненти з інтерфейсом без логіки, щоб ми могли їх перевикористати і масштабувати в більших компонентах з логікою.

Для нашого менеджменту станів додатки ми створимо окремо стор і обгорнемо головний компонент в стор, так ми зможемо отримувати стану компонентів і дані отримані сервера в будь-якому компоненті. Для сховища ми створимо папку модулів робіт станів де буде багато модулів і кожен буде працювати з певною сутностей і в головному модулі ми пов'язуємо всі модулі і передаємо в сховище. У самому модулі будуть описані дії з певною сутностей і Етій дій може бути безлічі, наприклад видалити, додати, відредагувати. Щоб розуміти яке дії викликати ми створюємо папку з діями і дамо їм типи потім за цими типами модулі станів розумітимуть яке дії викликати, типи ми оголошуємо теж окремо в константах.

Для роботи з сервером ми будемо використовувати бібліотеку Axios js а для того щоб ми могли дочекатися відповіді від сервера і зробити дії подальших з результатами ми будемо використовувати Redux-thunk js він нам допоможе створити асинхронні екшени щоб при відповіді з сервера ми змогли отримані дані відправити в модуль станів і зробити там розрахунки з ними, для цього ми створимо папку моделі і там будуть багато моделей для різних сутностей, в певній сутності у нас будуть три виклики дій по певній сутності перша це очікування відповіді від сервера другий це успішний результат від сервера і третій це не успішний відповідь від сервера і з кожним відповіддю ми будемо подальшу ланцюжок дій.

Подивіться на готовий результат архітектури на малюнку 4.1

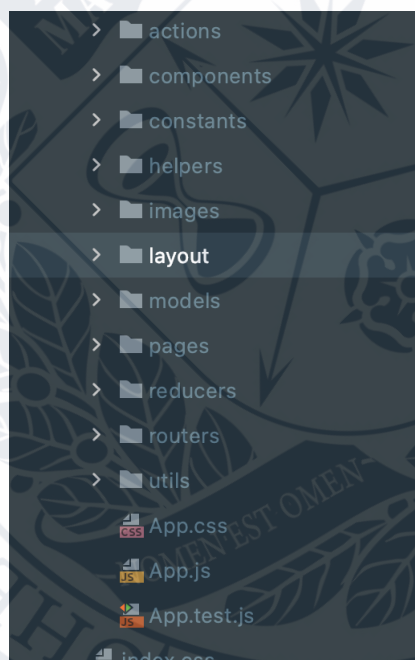


Рисунок 4.1 - Архітектура додатку

Для роботи з сервером ми побудуємо окремий клас де в одному місці опишемо принципи роботи з сервером, ми опишемо заголовки і токен, також ми опишемо як реагувати на помилки, щоб за будь-якої помилки ми додавали її в тостер і потім тостер завдяки тому що він компонент верхнього порядку відображається у нас всюди, так у нас користувач відразу ж розуміє що відбувається не перериваючи роботу з веб додатком, це все у нас буде в файлі httpClient.js

Потім ми зробимо клас де ми ініціюємо наш виклик класу для роботи з сервером і в ньому опишемо всі роботи із запитам для сервера, і потім будемо просто з цього класу викликати певний метод і він вже з усіма описаними характеристика раніше буде працювати, цей файл буде `mainApi.js`

Далі для певних станів в ізолюваних компонентах ми будемо використовувати рідні функції `React.js` це будуть `useState` для певних змінних щоб при зміні стані у нас оновився компонент і відображав нову інформацію. І для життєвих циклів компонента ми будемо використовувати `useEffect` він нам дасть можливість при певних моментах самому робити розрахунки і оновлювати компоненти.

Також для оптимізації і позбавлення від зайвих оновлень компонентів ми будемо використовувати такі функції як `useMemo` і `useCallback` вони нам допоможуть ізолювати певні розрахунок і не допускати щоб при зайвих оновлень компонента ці розрахунок зайвий раз не перераховувалися.

4.3 Готовий продукт та тестування

При завершенні проекту ми можемо подивитися на процеси взаємодії в веб додатку на малюнку 4.2

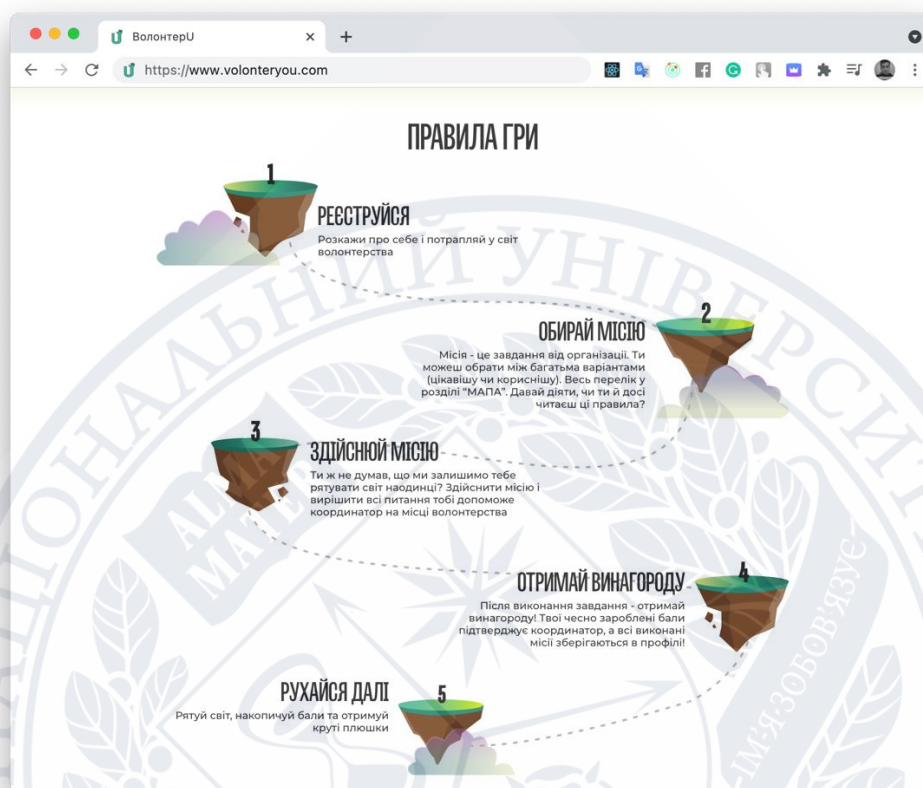


Рисунок 4.2 – Процеси роботи в веб додатку

У нас є напрямок для волонтерський місій малюнок 4.3

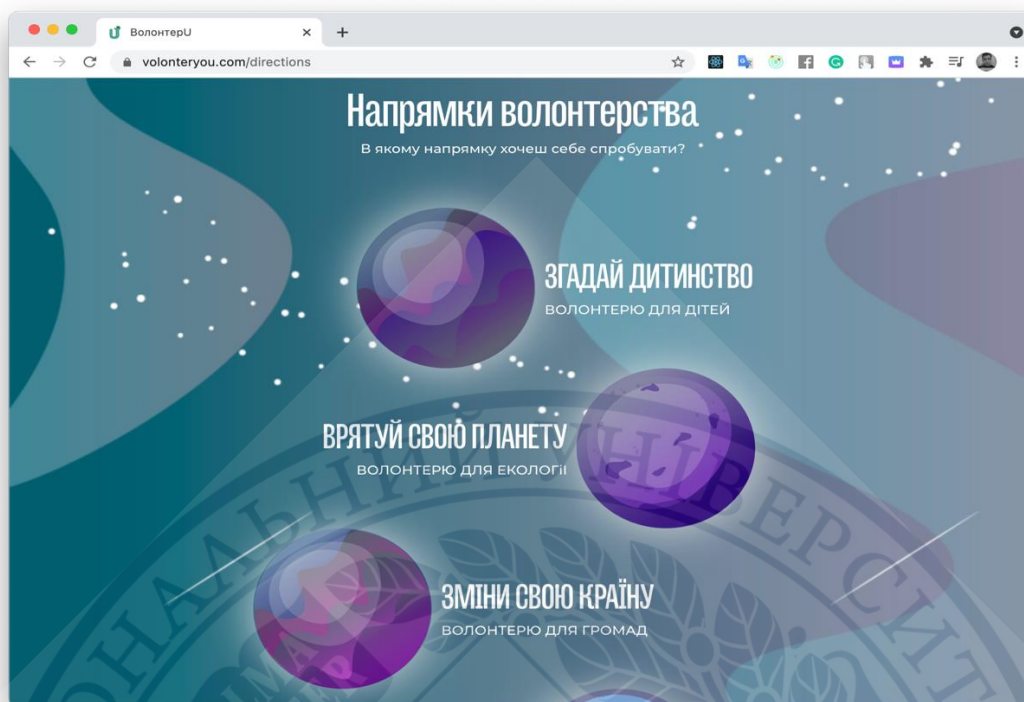


Рисунок 4.3 – Напрямки волонтерської діяльності

Також в кожному напрямку є місії з певної тематики, що допомагає сконцентруватися з проблеми на малюнках 4.4 і 4.5 ми можемо побачити список місій і можливість записатися на них.

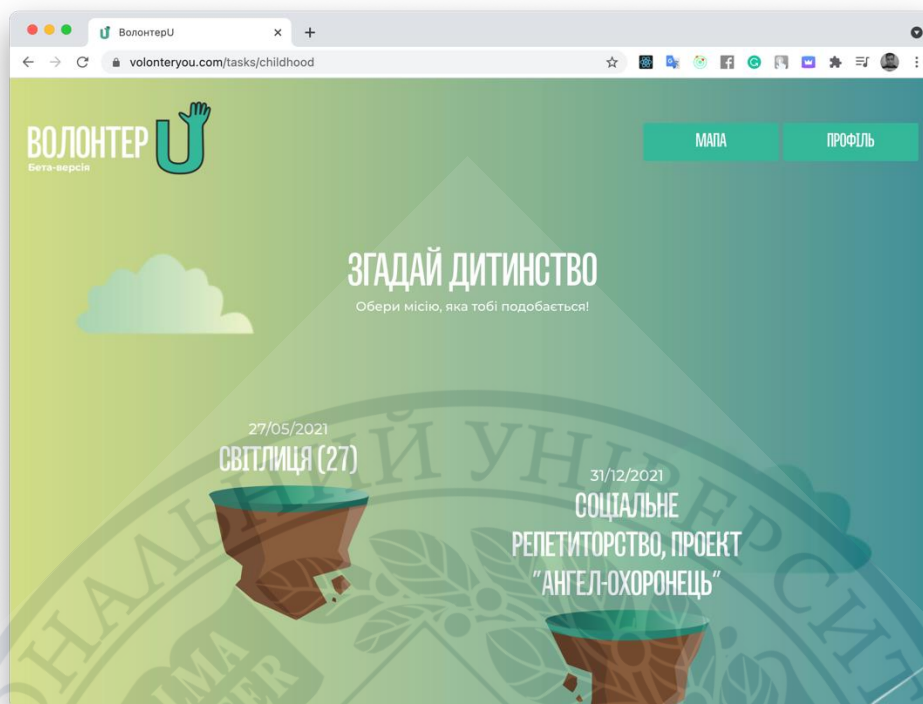


Рисунок 4.4 – Місії по певної тематики

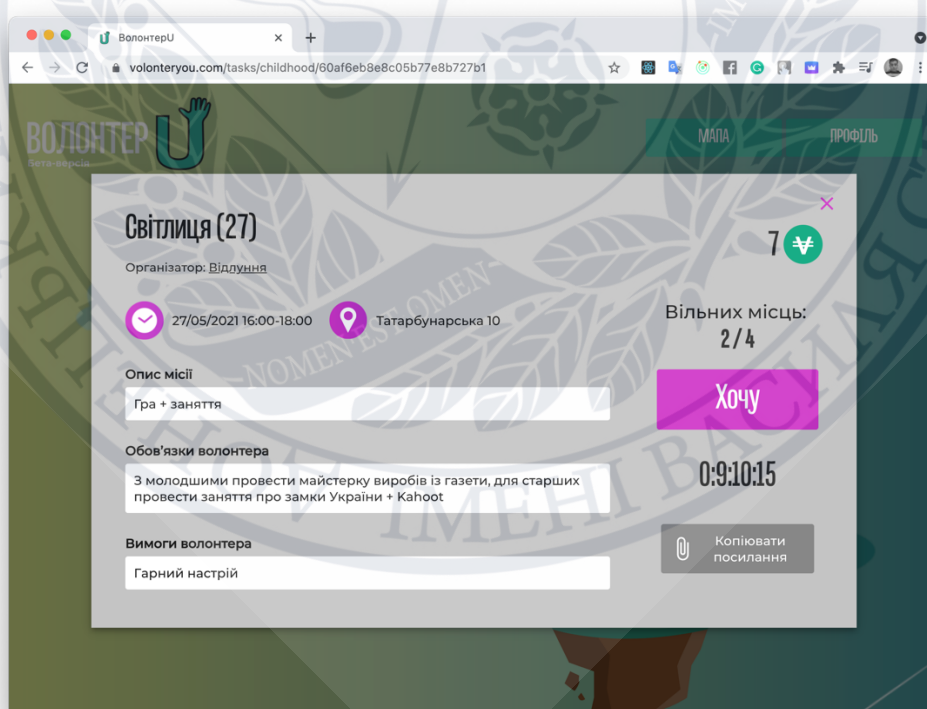


Рисунок 4.5 – Тут ми бачимо можливість дізнатися по місію та записатися на неї

Далі ми можемо в профілі взаємодіяти завдання і відстежувати їх виконання і стежити за процесом нагороди, також ми бачимо активні і виконані місії на малюнку 4.6

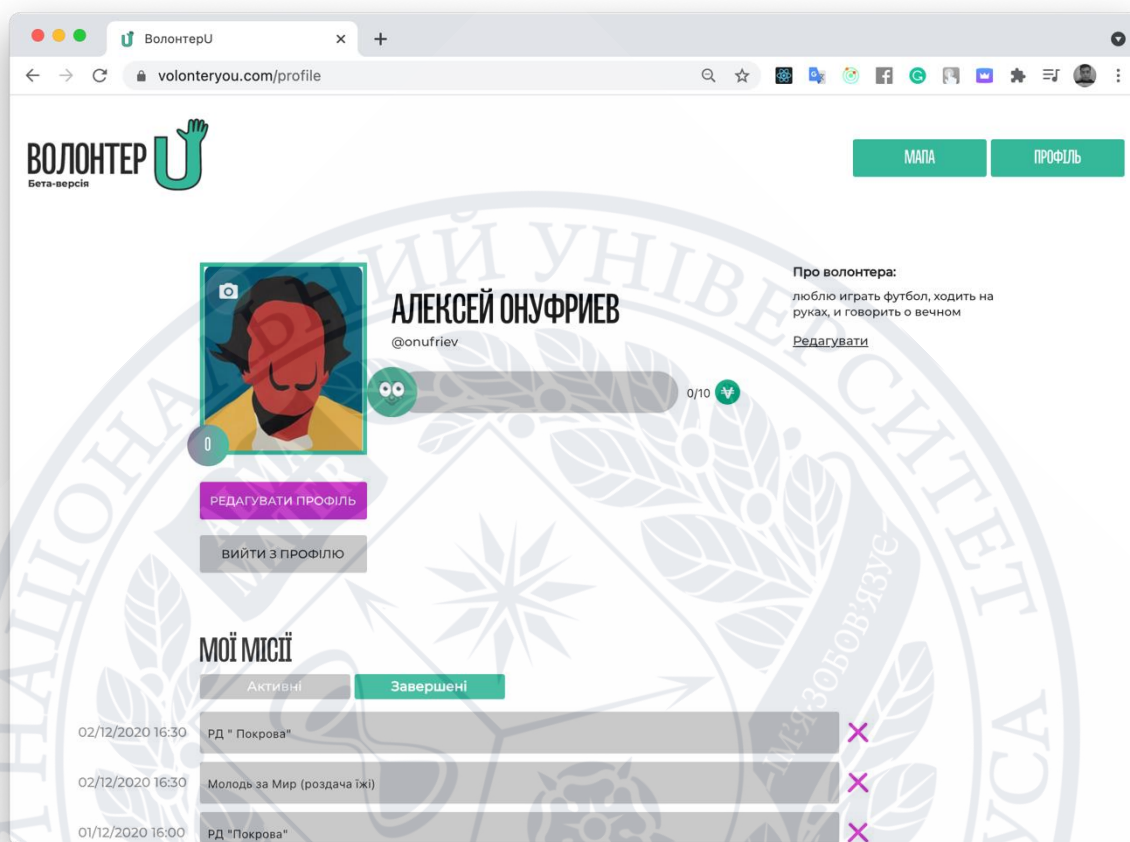


Рисунок 4.6 – Профіль користувача

Для організації зробили зручну форму створення місії де організація може описати всі критерії за якими набирає волонтерів? Тим самим волонтерам дає відразу зрозуміти опис місії кого шукають і що очікують від кандидатів. Що набагато спрощує і прискорює всі процеси, на малюнках 4.7 і 4.8 ми можемо побачити форму створення місії.

Створення місії

Назва місії*
Назва місії

Опис місії
Опис місії

Обов'язки волонтера*
Обов'язки волонтера

Вимоги до волонтера
Вимоги до волонтера (необхідні довідки, маски і т.д.)

Тип місії*
Тип місії

Дата*
May 27th

Час*

Рисунок 4.7 – Форма створення місії

May 27th

Час*
Час

Адреса*
Введіть адресу

Дедлайн реєстрації на місію
Дата*
May 27th

Час*
Час

Кількість балів за проходження місії*
5 7 15 25 50

Кількість волонтерів*
1 100

< 0 >

Вкажіть тривалість місії
ВКАЖІТЬ ТРИВАЛІСТЬ МІСІЇ

ЗАНЯТИ ДОДАТИ МІСІЮ

Рисунок 4.8 – Форма створення місії

ВИСНОВКИ

У висновку ми можемо підкреслити важливість волонтерської діяльності, що розвинена інфраструктура волонтерства покращує навколишнє середовище і життя навколо. У даній роботі ми проаналізували саме рух волонтерства як воно вирішує проблеми і допомагає людям, знайшли якісь стоять виклики серед них і зробили інструмент який допомагає і спрощує волонтерські процеси.

Веб додатком вже користуються волонтерські організації і вужчим є позитивний фідбек, набагато прискорює і спрощує роботу організаціям і волонтерам.

У самій розробці платформи ми використовували шаблони проектування, архітектурні рішення та новітні технології. Ми побудували швидке додатки яка базується на компонентному підході що прискорює роботу веб додатки, так само побачили випадки якщо ми хочемо масштабувати або змінити певний компонент то не потрібно зачіпати інші компоненти.

Також ми використовували метод шаблонного проектування Сінглтон для роботи з сервером, що дозволяє перевикористати код і описати всі критерії в одному місці.

ПЕРЕДІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Спогад про битву при Сольферіно: книга

URL: https://www.icrc.org/ru/doc/assets/files/2013/solferino_rus.pdf

2. Веб додаток МКЧХ: веб сайт.

URL: <https://compagnonsbatisseurs.world/>

3. Відлуння львівська благодійна організація : сторінка у соц-мережі.

URL: https://www.icrc.org/ru/doc/assets/files/2013/solferino_rus.pdf

4. Бельгійський веб додаток для волонтерської діяльності: веб сайт.

URL: <https://compagnonsbatisseurs.world/>

5. Книга про досвід користувача дизайнера: книга

URL: https://www.userfocus.co.uk/pdf/Bright_Ideas_for_UX_Designers.pdf

5. Книга про досвід користувача дизайнера: книга

URL: https://www.userfocus.co.uk/pdf/Bright_Ideas_for_UX_Designers.pdf

6. Книга про шаблони проектування на JavaScript: книга

URL: <https://addyosmani.com/resources/essentialjsdesignpatterns/book/>

7. React js бібліотека для проектування веб додатку: веб сайт

URL: <https://reactjs.org/>

8. Redux js бібліотека для контролю станів веб додатку: веб сайт

URL: <https://redux.js.org/>

9. Material UI бібліотека з готовими рішеннями для компонентів додатку: веб сайт

URL: <https://material-ui.com/>

9. Готова бакалаврська робота: веб сайт

URL: <https://www.volonteryou.com/>

ДОДАТОК

```
// App.js

import React from "react"
import {
  BrowserRouter,
} from "react-router-dom";
import {createStore, applyMiddleware, compose} from 'redux';
import {Provider} from 'react-redux';
import './App.css'
import thunk from 'redux-thunk'
import rootReducer from './reducers';

import Layout from './layout'

import 'semantic-ui-css/semantic.min.css'
import cacheImages from './utils/imageCaching';

// const ps = window.__REDUX_DEVTOOLS_EXTENSION__ &&
// window.__REDUX_DEVTOOLS_EXTENSION__()
export const store = createStore(rootReducer, compose(
  applyMiddleware(thunk),
  window.devToolsExtension ? window.devToolsExtension() : f => f
))
)

cacheImages();

export default function App() {
  return (
    <Provider store={store}>
      <BrowserRouter>
        <Layout />
      </BrowserRouter>
    </Provider>
  )
}
```

```
// layout/index.js

import React, {useState, useEffect} from "react"
import {
  Link,
  useLocation,
```

```

  Redirect
} from "react-router-dom"
import { useDispatch, useSelector } from 'react-redux'
import Loader from 'react-loader-spinner'
import axios from 'axios'
import Content from '../routers'
import {Header, Footer, ConfirmationWindow, Toasts} from '../components'
import {PAuth} from '../pages'
import {getMyUser} from '../models'
import {requestTest} from '../constants/request'
import { loadingTrue, loadingFalse } from '../actions/loading'
import {
  TAB_LENDING
} from '../constants/index'

import './index.css'

const Layout: React.FC = () => {
  const { pathname } = useLocation()
  const { user, loading } = useSelector((state: any) => state)
  const dispatch = useDispatch()
  const { authTab } = useSelector((state: any) => state)

  const wakeUpServer = () => {
    dispatch(loadingTrue())
    axios.get(requestTest)
      .then(() => {
        dispatch(loadingFalse())
      })
  }

  useEffect(() => {
    wakeUpServer()
    if (localStorage.getItem('token') && !user.email) {
      console.log(11111)
      dispatch(getMyUser())
    }
  }, [!!localStorage.getItem('token'), !user.email])

  const whereNotUse = [
    '/auth',
    '/profile-change',
  ]

```

```

const useWhiteLogo = [
  '/profile',
  '/profile/',
  '/volonter',
  '/organization'
]

const isCurrent = whereNotUse.indexOf(pathname) === -1
const isWhiteLogo = useWhiteLogo.indexOf(pathname) === -1

return loading ? (
  <div className='loader'>
    <Loader
      type="BallTriangle"
      color="#00BFFF"
      height={80}
      width={80} />
    </div>
  ) : (
    <div className='layout'>
      <Header isCurrent={isCurrent} isWhiteLogo={isWhiteLogo} />
      <div className='content'>
        <Content />
      </div>
      <Toasts />
      <ConfirmationWindow />
      {!isCurrent || authTab.content === TAB_LENDING && (<Footer />)}
    </div>
  )
}

export default Layout;
// routers/index.js
import React from 'react'
import {
  Switch,
  Route,
  Redirect
} from 'react-router-dom'
import {
  PDirections,
  PAuth,
  PProfileUser,
  PProfileUserChange,

```



```

PTasks,
PTasksNotAuth
} from '../pages'

export default function Content() {

  const isLogin = !!localStorage.getItem('token')

  return (
    <Switch>
      <Route path='/profile'>
        {isLogin ? <PProfileUser /> : <Redirect to='/auth' />}
      </Route>
      <Route path='/volonter'>
        {isLogin ? <PProfileUser /> : <Redirect to='/auth' />}
      </Route>
      <Route path='/organization'>
        {isLogin ? <PProfileUser /> : <Redirect to='/auth' />}
      </Route>
      <Route path='/profile-change'>
        {isLogin ? <PProfileUserChange /> : <Redirect to='/auth' />}
      </Route>
      <Route path='/tasks/:directionType/:taskId' component={isLogin ? PTasks :
PTasksNotAuth} />
      <Route path='/tasks/:directionType' component={isLogin ? PTasks : PAuth} />
      <Route path='/auth'>
        {!isLogin ? <PAuth /> : <Redirect to='/directions' />}
      </Route>
      <Route path='/directions'>
        {isLogin ? <PDirections /> : <Redirect to='/auth' />}
      </Route>
      <Route path='/'>
        {/*{isLogin ? <PDirections /> : <Redirect to='/auth' />}*/}
        <PAuth />
      </Route>
    </Switch>
  );
}

```

// Organization/index.js

```

import React, { useState, useRef, useEffect } from "react";
import { Link } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";

```

```

import { updateUser } from "../../models";
import { getMyUser } from "../../models";
import { Task, TextInput } from "../../components";
import { useLocation, useHistory } from "react-router-dom";
import DefaultAvatar from "../../images/avatar.png";
import { makeStyles } from "@material-ui/core/styles";
import { FETCH_USERS_CLEAR } from "../../constants/index";
import IconPlus from "../../images/icon-plus.svg";
import IconLocation from "../../images/icon-location.svg";
import IconButton from "@material-ui/core/IconButton";
import PhotoCameraIcon from "@material-ui/icons/PhotoCamera";
import ArrowBackRoundedIcon from '@material-ui/icons/ArrowBackRounded';
import IconPhone from "../../images/icon-phone.svg";
import IconMessage from "../../images/icon-message.svg";
import TextField from "@material-ui/core/TextField";
import { ModalCreateTask } from "../ModalCreateTask";
import { ModalTaskInfo } from "../ModalTaskInfo";
import { confirmCancel, confirmOk } from "../../actions/confirm";

import { useClearCache } from "react-clear-cache";

import axios from "axios";
import {
  requestCreateTask,
  requestGetTask,
  requestGetTaskOrg,
  requestUploadAvatar,
  requestDeleteTask,
  requestGetTaskVolunteers,
} from "../../constants/request";

import "../index.css";
import style from "../Organization.module.css";
import MainApi from "../../utils/mainApi";
import { pushToast } from "../../actions/toasts";
import Compress from "react-image-file-resizer";

const TYPE_UPCOMING = "upcoming";
const TYPE_COMPLETE = "completed";
const initLimit = 10;

const useStyles = makeStyles((theme) => ({
  modal: {
    display: "flex",
    alignItems: "center",
  },

```

```

    justifyContent: "center",
  },
  paper: {
    backgroundColor: theme.palette.background.paper,
    border: "2px solid #000",
    boxShadow: theme.shadows[5],
    padding: theme.spacing(2, 4, 3),
  },
  underline: {
    "&&:before": {
      borderBottom: "none",
    },
    "&&:after": {
      borderBottom: "none",
    },
  },
});

export default function Organization({ forceUpdate, user, seenByVolonter }) {
  const classes = useStyles();
  const mainApi = MainApi.getInstance();
  const history = useHistory();
  const location = useLocation();
  const [isEditDescriptionMode, setIsEditDescriptionMode] = useState(false);
  const dispatch = useDispatch();
  const refUpload = useRef();
  const { isLatestVersion, emptyCacheStorage } = useClearCache();
  const [description, setDescription] = useState(user?.description);
  const [descriptionValue, setDescriptionValue] = useState(description);
  const [error, setError] = useState("");
  const [avatar, setAvatar] = useState(user?.imgPath || DefaultAvatar);
  const [tasks, setTasks] = useState([]);
  const [visibleModalCreate, setVisibleModalCreate] = useState(false);
  const [visibleModalInfo, setVisibleModalInfo] = useState(location.state);
  const [currentTask, setCurrentTask] = useState();
  const [currentVolunteers, setCurrentVolunteers] = useState();
  const linkRef = useRef();
  const [missionTab, setMissionTab] = useState(TYPE_UPCOMING);
  const [limit, setLimit] = useState(initLimit);

  const logout = () => {
    const isLogout = window.confirm("Ви нас залишаєте?");
    if (isLogout) {
      dispatch({ type: FETCH_USERS_CLEAR });
      localStorage.clear();
    }
  };

```

```

    }
  };

```

```

const updateTasks = async () => {
  if(!seenByVolonter) {
    const res = await mainApi.getMissionForOrganization({
      orgId: user.id,
      status: missionTab,
      limit,
    });
    setTasks(res.data);
  }
};

```

```

const pickImg = async (e) => {
  setError("")
  const f = e.target.files[0]
  const fd = new FormData()

  Compress.imageFileResizer(
    f,
    400,
    430,
    "WEBP",
    100,
    0,
    (resizedFile) => {

      fd.append('file', resizedFile)

      uploadImg(fd).then(() => {

        setAvatar(URL.createObjectURL(resizedFile))

        dispatch(pushToast({
          status: 'success',
          text: 'Фото завантажено'
        })))

      }).catch(() => {

        dispatch(pushToast({
          status: 'error',
          text: 'Помилка завантаження зображення'
        })))

      })
    }
  )
}

```



```

    },
    'blob'
  );

  // if (f.size > 1000000) {
  //   return setError('розмір картинки занадто великий')
  // }
}

const uploadImg = async (fd) => {
  await axios({
    method: "post",
    url: requestUploadAvatar,
    data: fd,
    headers: {
      Authorization: localStorage.getItem("token"),
    },
  });
};

useEffect(() => {
  dispatch(getMyUser());
}, []);

useEffect(() => {
  updateTasks();
}, [missionTab, limit]);

const createTask = async (values) => {
  // todo

  const data = {
    organizationId: user.id,
    ...values,
  };

  await axios({
    method: "post",
    url: requestCreateTask,
    data,
    headers: {
      Authorization: localStorage.getItem("token"),
    },
  });
};

```

```

updateTasks();
setVisibleModalCreate(false);
};

```

```

const deleteTask = async (id) => {
  await axios({
    method: "delete",
    url: requestDeleteTask + "/" + id,
    headers: {
      Authorization: localStorage.getItem("token"),
    },
  });
};

```

```

updateTasks();
};

```

```

const removeTask = (id) => {
  dispatch(
    confirmOk({
      title: "ВИ ВПЕВНЕНІ, ЩО ХОЧЕТЕ ВИДАЛИТИ МІСІЮ?",
      open: true,
      errorAlert: false,
      onClose: () => {
        dispatch(confirmCancel());
      },
      onOk: () => {
        deleteTask(id);
        dispatch(confirmCancel());
        dispatch(pushToast({
          status: 'success',
          text: 'Місію скасовано!'
        })))
    },
  )
);
};

```

```

const updateDescription = () => {
  const changedData = { ...user };
  changedData.description = descriptionValue;
  dispatch(updateUser(changedData));
  setDescription(descriptionValue);
  setIsEditDescriptionMode(false);
};

```

```

const clickTask = async (data) => {
  setCurrentVolunteers(null);
  setCurrentTask(null);
  await mainApi
    .getVolontersByMission({
      missionId: data.id,
    })
    .then((res) => {
      setCurrentVolunteers(res.data);
      setCurrentTask(data);
      setVisibleModalInfo(true);
    })
    .catch((e) => {
      console.log(e.message);
    });
};

useEffect(() => {
  if(location.state?.isComeFromVolonter) {
    clickTask(location.state.mission)
  }
}, [])
return (
  <div className="cover c-profile p-profile-volunteer">
    <Link to="/profile-change" ref={linkRef} style={{ display: "none" }} />
    {
      seenByVolonter
        ? <div style={{margin: `10px 0 50px 150px`, display: 'flex', alignItems: 'center',
alignSelf: 'flex-start'}}
          onClick={() => history.push({
            pathname: location.state.missionPath,
          })}>
            <div type="button" className="back-button__volonter">
              <ArrowBackRoundedIcon classes={{root: 'zero-step__back-button-icon'}}
fontSize="inherit"/>
            </div>
            <span style={{marginLeft: 10, fontFamily: 'Montserrat', fontSize:
15}}>Повернутися до місії</span>
          </div>
          : null
        }
    }
    <div className="p-profile-content">
      <div className="p-profile-content-left">
        <div
          className="p-profile-div-avatar"

```

```

style={{ position: "relative" }}
>
<img className="p-profile-img-avatar" src={avatar} />
{seenByVolonter ? null : (
  <>
    <IconButton
      color="primary"
      component="span"
      classes={{ root: "avatar-photo__wrapper" }}
      onClick={() => refUpload.current.click()}
    >
      <PhotoCameraIcon style={{ color: "#fff", fontSize: 26 }} />
      <input
        ref={refUpload}
        style={{ display: "none" }}
        type="file"
        onChange={pickImg}
        accept="image/*"
      />
    </IconButton>
    <div
      className="div-button-edit-profile"
      onClick={() => linkRef.current.click()}
    >
      <p className="text white">РЕДАГУВАТИ ПРОФІЛЬ</p>
    </div>
    <br />
    <div className="div-button-logaut" onClick={logout}>
      <p className="text">ВИЙТИ з профілю</p>
    </div>
  </>
)}
</div>
<div className="p-profile-div-info-user">
  <p className={style.mainTitle}>{user.name}</p>
  <div
    style={{
      display: "flex",
      flexDirection: "row",
      alignItems: "center",
      marginBottom: 5,
    }}
  >
    <img
      src={IconLocation}

```



```

        style={{ width: 35, height: 35, marginRight: 5 }}
      />
      <p style={{ fontFamily: "Montserrat", fontSize: 15 }}>
        {user?.address}
      </p>
    </div>
    <div
      style={{
        display: "flex",
        flexDirection: "row",
        alignItems: "center",
        marginBottom: 5,
      }}
    >
      <img
        src={IconPhone}
        style={{ width: 35, height: 35, marginRight: 5 }}
      />
      <p style={{ fontFamily: "Montserrat", fontSize: 15 }}>
        {user?.phoneNumber}
      </p>
    </div>
    <div
      style={{
        display: "flex",
        flexDirection: "row",
        alignItems: "center",
        marginBottom: 5,
      }}
    >
      <img
        src={IconMessage}
        style={{ width: 35, height: 35, marginRight: 5 }}
      />
      <p style={{ fontFamily: "Montserrat", fontSize: 15 }}>
        {user?.email}
      </p>
    </div>
    <div className="p-profile-div-line-level">
      {/*<p>line level</p>*/}
    </div>
  </div>
</div>

<div className="p-profile-content-right">

```

```

<div className="p-profile-div-description">
  <p
    style={{
      fontFamily: "Montserrat",
      fontSize: 15,
      fontWeight: "bold",
      marginBottom: 10,
      marginLeft: 10,
    }}
  >
    Про організацію:
  </p>
  {isEditDescriptionMode ? (
    <TextField
      multiline
      rowsMax={10}
      rows={10}
      value={descriptionValue}
      onChange={(e) => setDescriptionValue(e.target.value)}
      InputProps={{ classes }}
      autoFocus
      style={{
        fontSize: 14,
        padding: "0 10px",
        borderRadius: 5,
        border: isEditDescriptionMode ? "1px solid #ebf9f6" : "none",
      }}
    />
  ) : (
    <p
      className="text"
      style={{
        textTransform: "none",
        padding: "0 10px",
        hyphens: "auto",
        maxHeight: 200,
        overflow: "auto",
      }}
    >
      {description}
    </p>
  )}
  {seenByVolonter ? null : isEditDescriptionMode ? (
    <div
      style={{

```

```

        display: "flex",
        justifyContent: "flex-start",
        marginTop: 10,
        marginLeft: 10,
    }}
>
<p
  className="edit-text__button"
  style={{ marginRight: 20 }}
  onClick={() => updateDescription()}
>
  Зберегти
</p>
<p
  className="edit-text__button"
  onClick={() => {
    setIsEditDescriptionMode(false);
    setDescriptionValue(description);
  }}
>
  Скасувати
</p>
</div>
): !isEditDescriptionMode ? (
<p
  className="edit-text__button"
  style={{ marginLeft: 10 }}
  onClick={() => setIsEditDescriptionMode(true)}
>
  Редагувати
</p>
): null}
</div>
</div>
</div>
{seenByVolonter ? null : (
  <div
    className="block-btn-plus"
    onClick={() => setVisibleModalCreate(true)}
  >
    <p
      style={{
        marginBottom: 0,
        marginRight: 5,
        fontWeight: "bold",

```

```

    }}
  >
    Створити місію
  </p>
  <img src={IconPlus} />
</div>
))

```

```

{seenByVolonter ? null : (
  <div className={style.blockMissions}>
    <p className={style.title}>МОЇ МІСІЇ</p>
    <div className={style.blockMissionButtons}>
      <div
        className={
          missionTab === TYPE_UPCOMING
            ? style.missionTabActive
            : style.missionTab
        }
        onClick={() => setMissionTab(TYPE_UPCOMING)}
      >
        <p
          className={
            missionTab === TYPE_UPCOMING
              ? style.missionTextActive
              : style.missionText
          }
        >
          АКТИВНІ
        </p>
      </div>
      <div
        className={
          missionTab === TYPE_COMPLETE
            ? style.missionTabActive
            : style.missionTab
        }
        style={{
          backgroundColor: missionTab !== TYPE_UPCOMING && "#35B899",
        }}
        onClick={() => setMissionTab(TYPE_COMPLETE)}
      >
        <p
          className={
            missionTab === TYPE_COMPLETE
              ? style.missionTextActive

```



```

        : style.missionText
      }
    >
    Завершені
  </p>
</div>
</div>
{tasks.length > 0 && (
  <div className={style.taskList}>
    {tasks.map((mission) => {
      return (
        <Task
          key={mission.id}
          mission={mission}
          deleteTask={() => removeTask(mission.id)}
          onClick={() => clickTask(mission)}
        />
      );
    })}
  </div>
)}
{limit === tasks.length && (
  <div
    className={style.missionTabActive}
    onClick={() => setLimit(limit + initLimit)}
  >
    <p className={style.missionTextActive}>Шукати ще місії +</p>
  </div>
)}
</div>
)}

<ModalCreateTask
  open={visibleModalCreate}
  handleClose={() => setVisibleModalCreate(false)}
  createTask={createTask}
/>
{currentTask && (
  <ModalTaskInfo
    open={visibleModalInfo}
    handleClose={() => setVisibleModalInfo(false)}
    handleOpen={() => setVisibleModalInfo(true)}
    currentTask={currentTask}
    updateTasks={updateTasks}
    currentVolunteers={currentVolunteers}
  />
)}

```

```

        deleteTask={deleteTask}
      />
    )}
  </div>
);
}

// utils/httpClient.js
// @ts-nocheck
/* eslint-disable */

import axios, {
  AxiosInstance,
  AxiosResponse,
  AxiosRequestConfig,
  AxiosError,
} from 'axios';
import { store } from '../App';
import { pushToast } from '../actions/toasts';
import apiRes from '../api-responses.json';
import { logout } from '../actions/login';

abstract class HttpClient {
  protected readonly instance: AxiosInstance;
  protected accessToken = localStorage.getItem('token') || '';

  public constructor(baseUrl: string) {
    this.instance = axios.create({
      baseUrl: baseUrl,
      headers: {
        Authorization: `Bearer ${this.accessToken}`
      },
    });
  };

  this.instance.interceptors.request.use(this._handleRequest, this._handleError);
  this.instance.interceptors.response.use(this._handleResponse, this._handleError);
}

private _handleRequest = (config: AxiosRequestConfig): AxiosRequestConfig => {
  Object.assign(config.headers, {
    Authorization: `Bearer ${localStorage.getItem('token') || ''}`,
  });
  const tokenVal = localStorage.getItem('tokenStartTime') || '';

  if (tokenVal) {

```

```

const oldTime = new Date(tokenVal);
const newTime = new Date();

oldTime.setDate(oldTime.getDate() + 14);

if (oldTime.getTime() < newTime.getTime()) {
  localStorage.setItem('token', '');
  localStorage.setItem('tokenStartTime', '');
  store.dispatch(logout())
}
}

if (
  config.url &&
  config.url.indexOf('refresh') === -1 &&
  config.url.indexOf('login') === -1
) {
  // const tokenVal = localStorage.getItem('tokenStartTime') || '';
  // const time = new Date(tokenVal);
  // if (
  //   tokenVal !== '' &&
  //   new Date() > new Date(time.setMinutes(time.getMinutes() + 50))
  // ) {
  //   this.refreshAccessToken().then((response) => {
  //     localStorage.setItem('token', response.data.data.attributes.access_token);
  //     localStorage.setItem('tokenStartTime', new Date().toUTCString());
  //   });
  // }
  return config;
};

private _handleResponse = (
  response: AxiosResponse
): AxiosResponse | Promise<AxiosResponse> => {
  return response;
};

private _handleError = (error: AxiosError) => {
  if (error.response) {
    let isShow = true;
    if (store.getState().toasts?.length > 0) {
      const texts = store.getState().toasts.map(el => el?.text) || []
      const index =
        texts.indexOf(apiRes[error.response.status.toString()][error.response.data.messageLa

```

```

bel))

    if (index !== -1) {
        isShow = false;
    }
}

if (isShow) {
    store.dispatch(pushToast({
        status: 'error',
        text:
apiRes[error.response.status.toString()][error.response.data.messageLabel],
    )))
}
}

return Promise.reject(error.response);
}

private refreshAccessToken = async () => {
    return await this.instance.post<any>('auth/refresh', {}, {});
};
}

export default HttpClient;
// utils/mainApi.js
import { AxiosResponse } from 'axios';
import HttpClient from './httpClient';
import { BASE_URL } from './constants/request';

class MainApi extends HttpClient {
    private static classInstance?: MainApi;

    private constructor() {
        super(BASE_URL || "");
    }

    public static getInstance(): MainApi {
        if (!this.classInstance) {
            this.classInstance = new MainApi();
        }

        return this.classInstance;
    }
}

```



```

public async login(data: any): Promise<AxiosResponse<any>> {
    return await this.instance.post<any>('/login', data);
}

public async getMyUser(): Promise<AxiosResponse<any>> {
    return await this.instance.get<any>('/api/0.1/users');
}

public async getUserById({userId}: any): Promise<AxiosResponse<any>> {
    return await this.instance.get<any>('/api/0.1/users/' + userId);
}

public async getMissionForOrganization({status, limit = 10}: any):
Promise<AxiosResponse<any>> {
    return await this.instance.get<any>('/api/0.1/missions/' + status + '?limit=' + limit);
}

public async getAllMission({direction = "", limit = 10}: any):
Promise<AxiosResponse<any>> {
    return await this.instance.get<any>('/api/0.1/missions/?direction=' + direction +
    '&limit=' + limit);
}

public async getMissionForUser({status, limit = 10}: any):
Promise<AxiosResponse<any>> {
    return await this.instance.get<any>('/api/0.1/user-missions/' + status + '?limit=' +
    limit);
}

public async requestFinishTask(taskId: string, selectVol: any):
Promise<AxiosResponse<any>>{
    return await this.instance.patch<any>(`/api/0.1/missions/${taskId}`, selectVol)
}

public async deleteVolonterFromMission(userId: string, missionId: string):
Promise<AxiosResponse<any>> {
    return await this.instance.delete<any>(`/api/0.1/user-
missions?userId=${userId}&missionId=${missionId}`)
}

public async getCertainUser(userId: string): Promise<AxiosResponse<any>> {
    return await this.instance.get<any>(`/api/0.1/users/${userId}`)
}

public async getCertainMission(missionId: string): Promise<AxiosResponse<any>>

```

```

{
  return await this.instance.get<any>(`/api/0.1/missions/${missionId}/volunteers`)
}

public async getMissionById({taskId}: any): Promise<AxiosResponse<any>> {
  return await this.instance.get<any>(`/api/0.1/missions/` + taskId);
}

public async joinToTask({missionId}: any): Promise<AxiosResponse<any>> {
  return await this.instance.post<any>(`/api/0.1/user-missions/` + missionId);
}

public async getVolontersByMission({missionId}: any):
Promise<AxiosResponse<any>> {
  return await this.instance.get<any>(`/api/0.1/missions/` + missionId + `/volunteers`);
}

public async rejectMission( missionId: string): Promise<AxiosResponse<any>> {
  return await this.instance.delete<any>(`/api/0.1/user-missions/` + missionId);
}

public async changePassword(data: any): Promise<AxiosResponse<any>> {
  return await this.instance.patch<any>(`/api/0.1/users/pwd`, data);
}
}

export default MainApi;

// pages/PDirections/index.js
import React, { useEffect } from 'react'
import { useSelector } from 'react-redux'
import {
  useHistory
} from "react-router-dom"

import './index.css'
import axios from 'axios'

const directionName = {
  'CHILDHOOD': 'childhood',
  'ECOLOGY': 'ecology',
  'HOMELAND': 'homeland',
  'ANIMALS': 'animals',
  'INCLUSION': 'inclusion',
  'EVENTS': 'events'
}

```

```

}

export default function PDirections() {
  const { directions } = useSelector(state => state)
  const history = useHistory()

  const openDirection = (type) => {
    history.push('/tasks/' + directionName[type])
  }

  console.log(directions)
  return (
    <div className='cover c-directions p-directions'>
      <h1 className="direction-planets__title">Напрямки волонтерства</h1>
      <p className="direction-planets__subtitle">В якому напрямку хочеш себе
спробувати?</p>
      {
        directions.map((planet, index) => {
          return (
            <div
              className={'planet__item' + index}
              style={{ flexDirection: index % 2 === 0 ? 'row' : 'row-reverse', display: 'flex',
                alignItems: 'center',
                cursor: 'pointer', width: '100%', }}
              onClick={() => {
                console.log(planet.type)
                openDirection(planet.type)
              }}
            >
              <img className='img-planet-direction' src={planet.img}/>
              <div>
                <p className='planet__item-title white' style={{ textAlign: index % 2 ===
0 ? 'left' : 'right' }}>{planet.title}</p>
                <p className='planet__item-text white' style={{ textAlign: index % 2 ===
0 ? 'left' : 'right' }}>{planet.description}</p>
              </div>
            </div>
          )
        })
      }
    </div>
  );
}

```

```

// pages/PTasks/index.js
import React, {useState, useEffect, memo} from 'react'
import { useDispatch, useSelector } from 'react-redux'
import {
  Link,
  useLocation,
  Redirect,
  matchPath,
  useHistory
} from "react-router-dom"
import { withRouter } from "react-router";

import './index.css'
import styles from './PTask.module.css'
import axios from "axios";
import { getRandomIsland } from '../helpers'
import { getMyUser } from '../models'
import { makeStyles } from "@material-ui/core/styles";
import { ModalJoinTask } from './ModalJoinTask';
import MainApi from '../utils/mainApi';

import { joinToTask, requestGetUserById } from '../constants/request'
import { pushToast } from "../actions/toasts";

const initLimit = 10;

const directionName = {
  'childhood': {
    type: 'CHILDHOOD',
    title: 'ЗГАДАЙ ДИТИНСТВО'
  },
  'ecology': {
    type: 'ECOLOGY',
    title: 'ВРЯТУЙ СВОЮ ПЛАНЕТУ'
  },
  'homeland': {
    type: 'HOMELAND',
    title: 'ЗМІНИ СВОЮ КРАЇНУ'
  },
  'animals': {
    type: 'ANIMALS',
    title: 'ДАЙ ЛАПУ'
  },

```



```

'inclusion': {
  type: 'INCLUSION',
  title: 'ТВОРИ ДОСТУПНІСТЬ'
},
'events': {
  type: 'EVENTS',
  title: 'ПРОКАЧАЙСЯ'
}
}

```

```

const useStyles = makeStyles((theme) => ({
  modal: {
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'center',
  },
  paper: {
    backgroundColor: theme.palette.background.paper,
    border: '2px solid #000',
    boxShadow: theme.shadows[5],
    padding: theme.spacing(2, 4, 3),
    width: 300,
    height: 200
  },
}));

```

```

function PTasks(props) {
  const mainApi = MainApi.getInstance();
  const classes = useStyles();
  const dispatch = useDispatch()
  const [tasks, setTasks] = useState([])
  const {directionType} = props.match.params;
  const taskId = props.match?.params?.taskId;
  const [limit, setLimit] = useState(initLimit);
  const history = useHistory()

```

```

const updateTasks = () => {
  mainApi.getAllMission({direction: directionName[directionType].type, limit})
  .then((res) => {
    const readyMisson = res.data.map(mission => {
      mission.img = getRandomIsland()
      return mission
    })
    setTasks(readyMisson)
  })
}

```

```

}

useEffect(() => {
  dispatch(getMyUser())
}, [])

useEffect(() => {
  if (directionName[directionType]) {
    updateTasks()
  }
}, [limit])

const [currentTask, setCurrentTask] = useState()
const [currentOrg, setCurrentOrg] = useState()
const [open, setOpen] = useState(false);

const handleOpen = (missionId) => {
  history.push('/tasks/' + directionType + '/' + missionId)
  getMission(missionId)
};

const getMission = async (missionId) => {
  await mainApi.getMissionById({taskId: missionId})
  .then((res) => {
    console.log(res, 'res ----')
    console.log(res.data, 'res.data---')
    setCurrentTask(res.data)
    getOrganization(res.data)
  })
  .catch((e) => {
    console.log(e.message)
  })
}

const getOrganization = async (mission) => {
  await mainApi.getUserById({userId: mission.organizationId})
  .then((res) => {
    setCurrentOrg(res.data)
    setOpen(true);
  })
  .catch((e) => {
    console.log(e.message)
  })
}

```

```

const handleClose = () => {
  setOpen(false);
  history.push('/tasks/' + directionType)
};

const join = async (missionId) => {
  await mainApi.joinToTask({missionId})
  .then(() => {
    dispatch(pushToast({
      status: 'success',
      text: 'Круто! Тебе зараховано на цю місію! Відмічай нас в своїх
соцмережах! @volonteryou'
    })))
    handleClose()
  })
  .catch(() => {
    handleClose()
  })
}

const getTasks = () => {
  if (tasks.length === 0) return null;

  let array = tasks;
  let size = 3;
  let subarray = [];
  for (let i = 0; i < Math.ceil(array.length/size); i++){
    subarray[i] = array.slice((i*size), (i*size) + size);
  }

  return subarray.map((isles) => {
    return (
      <div className={styles.blockIsles} style={isles.length === 1 ? {width: '50%'} :
    {}>
      {
        isles.length > 0 && isles.map((isle, index) => {

          const IconIsle = isle.img;
          const isMove = index === 1;

          return (
            <div key={isle.id} className={isMove ? styles.movedIsle : styles.isle}
onClick={()=>handleOpen(isle.id)}>
              <p className={styles.dateIsle}>{new
Date(isle?.startDateTime).toLocaleDateString('en-GB')}</p>

```

```

        <p className={styles.titleIsle}>{isle?.name}</p>
        <IconIsle />
      </div>
    )
  })
}
</div>
)
})
}

useEffect(() => {
  if(taskId) {
    handleOpen(taskId);
  }
}, [])

if (!directionName[directionType]) return null

return (
  <div className='cover c-organizations p-tasks'>
    <h1 className={styles.title}>{directionName[directionType].title}</h1>
    <p className={styles.subtitle}>Обери місію, яка тобі подобається!</p>
    <div className={styles.blockTasks}>
      {
        getTasks()
      }
    </div>
    {limit === tasks.length && (
      <div className={styles.missionButton} onClick={() => setLimit(limit +
initLimit)}>
        <p className={styles.missionTextButton}>Шукати ще місії +</p>
      </div>
    )}

    {currentTask && (
      <ModalJoinTask
        open={open}
        handleClose={handleClose}
        currentTask={currentTask}
        currentOrg={currentOrg}
        join={() => join(currentTask.id)}
      />
    )}
  </div>

```



```
);  
}
```

```
export default memo(PTasks)
```



```
// PProfileUserChange/index.js
```

```
/*  
  todo: все названные классы тут дергают стили с компонент PAuth ->  
  RegistrationTab
```

todo: обновление, переписуем на изолированные стили и от предыдущего коментаря скоро избавимся

*/

```
import React, {useRef, useState} from 'react'
import {
  Link
} from "react-router-dom"
import { useDispatch, useSelector } from 'react-redux'
import { DatePickerInput, TextInput } from '../components'
import { Formik } from 'formik'

import axios from 'axios'
import {
  fetchUserReuest,
  fetchUserSuccess,
  fetchUserFailure
} from '../actions/login'
import { getMyUser, updateUser } from '../models'
import { pushToast } from '../actions/toasts';

import './index.css'
import * as Yup from "yup";
import moment from "moment";
import styles from './styles.module.css';
import MainApi from '../utils/mainApi';

const TABS = {
  GENERAL_DATA: 'generalData',
  CHANGE_PASSWORD: 'changePassword',
};

export default function PProfileUserChange () {
  const [tab, setTab] = useState(TABS.GENERAL_DATA);

  const {user} = useSelector(state => state)
  const dispatch = useDispatch()

  const linkRef = useRef()

  const phoneRegex = /^+\d{2}\s(\d{3})\s\d{3}\s\d{4}/;

  const save = (data) => {
    const mainApi = MainApi.getInstance();
```

```

switch(tab) {
  case TABS.CHANGE_PASSWORD: {

    const { currentPwd, newPwd } = data;

    mainApi.changePassword({ currentPwd, newPwd }).then(() => {
      dispatch(pushToast({
        status: 'success',
        text: 'Зміни збережено'
      }))
      linkRef.current.click()
    });

    break;
  }

  case TABS.GENERAL_DATA: {

    if (user.userType === 'ORGANIZATION') {
      data.contactPerson = {
        name: data.contactName,
        phoneNumber: data.phoneNumber
      }
    }

    if (user.userType === 'VOLUNTEER') {
      const dateUser = data.birthDate

      const year = moment(dateUser).format('yyyy');
      const month = moment(dateUser).format('MM');
      const newDay = moment(dateUser).format('DD');

      const readyDate = year + '-' + month + '-' + newDay;

      data.birthDate = readyDate
    }

    dispatch(updateUser(data))
    .then(() => {
      dispatch(getMyUser())
    })
    .then(() => {
      dispatch(pushToast({

```

```

        status: 'success',
        text: 'Зміни збережено'
    )))
    linkRef.current.click()
  })

  break;
}

default:
  break;
}
}

const initDataFirstStep = {
  id: user.id,
  firstName: user.firstName || "",
  lastName: user.lastName || "",
  // email: user.email || "",
  nickname: user.nickname || "",
  birthDate: user.birthDate || "",
  phoneNumber: user.phoneNumber || "",
  name: user?.name || "",
  address: user?.address || "",
  description: user.description || "",
  contactName: user?.contactPerson?.name || "",
  socialUri: user?.socialUri || ""
}

let validationDate = {
  phoneNumber: Yup.string()
    .matches(phoneRegExp, 'Номер телефону неправильний')
    .min(10, 'Повинен бути не менше 10 символів')
    .max(20, 'Повинен бути не більше 20 символів')
    .required('Напишіть номер телефону'),
  description: Yup.string()
    .max(1000, 'Повинен бути не більше 1000 символів')
    .required('Розкажіть про себе'),
  // email: Yup.string()
  //   .email('Пошта неправильна')
  //   .required('Напишіть пошту'),
}

if (user.userType === 'VOLUNTEER') {
  validationDate.firstName = Yup.string()

```



```

    .max(15, 'Повинен бути не більше 15 символів')
    .required("Напишіть ім'я");
validationDate.lastName = Yup.string()
    .max(20, 'Повинен бути не більше 20 символів')
    .required('Напишіть Прізвище');
validationDate.nickname = Yup.string()
    .max(20, 'Повинен бути не більше 20 символів')
    .required('Напишіть Нікнейм');
validationDate.birthDate = Yup.string()
    .required('Напишіть дату народження');
}

if (user.userType === 'ORGANIZATION') {
    validationDate.name = Yup.string()
        .max(15, 'Повинен бути не більше 15 символів')
        .required("Напишіть ім'я");
    validationDate.address = Yup.string()
        .max(200, 'Повинен бути не більше 200 символів')
        .required("Напишіть адресу");
    validationDate.contactName = Yup.string()
        .max(200, 'Повинен бути не більше 200 символів')
        .required("Напишіть ім'я контактної особи");
    validationDate.socialUri = Yup.string()
        .max(200, 'Повинен бути не більше 200 символів')
        .required("Напишіть послання на веб-сайт");
}

if (tab === TABS.CHANGE_PASSWORD) {
    validationDate.currentPwd = Yup.string()
        .min(9, 'Повинен бути не більше 200 символів')
        .required("Пароль має містити більше, ніж 9 знаків");

    validationDate.newPwd = Yup.string()
        .min(9, 'Повинен бути не більше 200 символів')
        .required("Пароль має містити більше, ніж 9 знаків");

    validationDate.repeatNewPwd = Yup.string()
        .min(9, 'Повинен бути не більше 200 символів')
        .required("Пароль має містити більше, ніж 9 знаків")
        .oneOf([Yup.ref('newPwd'), null], 'Паролі не співпадають');
}

const validation = Yup.object(validationDate)

return (

```

```

<div className='cover c-login'>
  <div style={{ flexDirection: 'row', display: 'flex', marginTop: 100,
    marginBottom: 100, width: '100%' }}>
    <div style={{ flex: 0.25, height: 'auto', alignItems: 'center', justifyContent: 'flex-
end', display: 'flex' }}>
      <div className={styles.tabs}>
        <button className={styles.tabs__item} onClick={() =>
setTab(TABS.GENERAL_DATA)}>
          <p className={styles.tabs__text + ' ' + (tab === TABS.GENERAL_DATA ?
styles.text_bold : '')}>Загальні дані</p>
        </button>
        <button className={styles.tabs__item} onClick={() =>
setTab(TABS.CHANGE_PASSWORD)}>
          <p className={styles.tabs__text + ' ' + (tab ===
TABS.CHANGE_PASSWORD ? styles.text_bold : '')}>Зміна паролю</p>
        </button>
      </div>
    </div>
    <div style={{ flex: 0.5, height: 'auto', display: 'flex', alignItems: 'center',
justifyContent: 'center' }}>
      <Formik
        initialValues={initDataFirstStep}
        onSubmit={(values, { setSubmitting }) => {
          save(values)
          setSubmitting(false)
        }}
        validationSchema={validation}
      >
        {({
          setFieldValue,
          values,
          handleSubmit,
          handleChange,
          handleBlur,
          errors,
          isSubmitting,
        }) => (
          <div style={{
            display: "flex",
            flexDirection: "column",
            justifyContent: "center",
            alignItems: "center",
            width: '438px',
          }}>

```

```

<p className={styles.title}>РЕДАГУВАННЯ ПРОФІЛЮ</p>
{
  tab === TABS.GENERAL_DATA && (
    <>
      {user.userType === 'VOLUNTEER' && (
        <>
          <TextInput
            style={{
              marginBottom: 18,
            }}
            value={values.firstName}
            onChange={handleChange('firstName')}
            onBlur={handleBlur('firstName')}
            error={errors.firstName}
            showTextError={true}
            placeholder='Ім'я'
          />
          <TextInput
            style={{
              marginBottom: 18
            }}
            value={values.lastName}
            onChange={handleChange('lastName')}
            onBlur={handleBlur('lastName')}
            error={errors.lastName}
            showTextError={true}
            placeholder='Прізвище'
          />
          <TextInput
            style={{
              marginBottom: 18
            }}
            value={values.nickname}
            onChange={handleChange('nickname')}
            onBlur={handleBlur('nickname')}
            error={errors.nickname}
            showTextError={true}
            placeholder='Нікнейм'
          />
          <DatePickerInput
            style={{
              marginBottom: 18
            }}
            value={values.birthDate}
            onChange={(data) => setFieldValue('birthDate', data)}

```

```

    onBlur={handleBlur('birthDate')}
    error={errors.birthDate}
    showTextError={true}
    placeholder='Дата народження'
  />
</>
)}
{user.userType === 'ORGANIZATION' && (
  <
    <TextInput
      style={{
        marginBottom: 18
      }}
      value={values.name}
      onChange={handleChange('name')}
      onBlur={handleBlur('name')}
      error={errors.name}
      showTextError={true}
      placeholder='Назва організації'
    />
    <TextInput
      style={{
        marginBottom: 18
      }}
      value={values.address}
      onChange={handleChange('address')}
      onBlur={handleBlur('address')}
      error={errors.address}
      showTextError={true}
      placeholder='Адреса'
    />
    <TextInput
      style={{
        marginBottom: 18
      }}
      value={values.contactName}
      onChange={handleChange('contactName')}
      onBlur={handleBlur('contactName')}
      error={errors.contactName}
      showTextError={true}
      placeholder="Ім'я контактної особи"
    />
    <TextInput
      style={{
        marginBottom: 18

```



```

    }}
    value={ values.socialUri }
    onChange={ handleChange('socialUri') }
    onBlur={ handleBlur('socialUri') }
    error={ errors.socialUri }
    showTextError={ true }
    placeholder="Послання на веб-сайт"
  />
</>
)}
<TextInput
  style={{
    marginBottom: 18,
  }}
  type='tel'
  value={ values.phoneNumber }
  onChange={ handleChange('phoneNumber') }
  onBlur={ handleBlur('phoneNumber') }
  error={ errors.phoneNumber }
  showTextError={ true }
  placeholder='Номер телефону'
/>
</>
)
}
{
  tab === TABS.CHANGE_PASSWORD && (
    <
    <TextInput
      style={{
        marginBottom: 18,
      }}
      value={ values.currentPwd }
      onChange={ handleChange('currentPwd') }
      onBlur={ handleBlur('currentPwd') }
      placeholder='Старий пароль'
      type='password'
      error={ errors.currentPwd }
      showTextError={ true }
    />
    <TextInput
      style={{
        marginBottom: 18,
      }}
      value={ values.newPwd }

```

```

        onChange={handleChange('newPwd')}
        onBlur={handleBlur('newPwd')}
        placeholder='Новий пароль'
        type='password'
        error={errors.newPwd}
        showTextError={true}
      />
      <TextInput
        value={values.repeatNewPwd}
        onChange={handleChange('repeatNewPwd')}
        onBlur={handleBlur('repeatNewPwd')}
        placeholder='Повторення нового паролю'
        type='password'
        error={errors.repeatNewPwd}
        showTextError={true}
      />
    </>
  )
}
{/* <TextInput
multiline
value={values.description}
onChange={handleChange('description')}
onBlur={handleBlur('description')}
error={errors.description}
showTextError={true}
placeholder='Про себе'
/> */}
{/*<TextInput*/}
{/* value={values.email}*/}
{/* onChange={handleChange('email')}*/}
{/* onBlur={handleBlur('email')}*/}
{/* error={errors.email}*/}
{/* showTextError={true}*/}
{/* placeholder='Пошта'*/}
{/*/>*/}
{/*<br />*/}
<div className={styles.block_buttons}>
  <button className={styles.button + ' ' + styles.bg_gray} onClick={() =>
linkRef.current.click()}>
    <p className={styles.button__title + ' ' +
styles.text_gray}>ЗАКРИТИ</p>
  </button>
  <button disabled={isSubmitting} className={styles.button + ' ' +
styles.bg_white} onClick={handleSubmit}>

```

```

        <p className={styles.button__title + ' ' +
styles.text_black}>ЗБЕРЕГТИ</p>
      </button>
    </div>
  </div>
)}
</Formik>
</div>
</div>
</div>
)
}

```

```
// auth/index.js
```

```

import React, {useState} from 'react'
import { useDispatch, useSelector } from 'react-redux'
import Lending from './Lending'

```

```

import LoginTab from './LoginTab'
import RegistrationTab from './RegistrationTab'
import ResetPW from './ResetPW'
import {
  setTabRegistration, setTabResetPW,
} from '../actions/setTab'
import {
  TAB_LENDING,
  TAB_LOGIN,
  TAB_REGISTRATION,
  TAB_RESET_PW
} from '../constants/index'
import { useHistory, useParams, useLocation } from "react-router-dom";

import './index.css'

export default function PAuth() {
  const isLogin = !!localStorage.getItem('token')
  const { authTab } = useSelector(state => state)
  const dispatch = useDispatch()
  const history = useHistory()
  const { search } = useLocation()
  // console.log(search, 'searchsearchsearchsearch')

  if (search.indexOf('?token=') !== -1) {
    if (authTab.content !== TAB_RESET_PW) {
      dispatch(setTabResetPW())
    }
  }

  const theToken = search.slice(7, search.length)

  return (
    <div className='cover c-login'>
      <ResetPW token={theToken}/>
    </div>
  )
}

switch (authTab.content) {
  case TAB_LENDING:
    return (<Lending setTab={() => {
      if (isLogin) {
        history.push('/directions')
      }
    }} />)
  case TAB_LOGIN:
    return (<LoginTab />)
  case TAB_REGISTRATION:
    return (<RegistrationTab />)
  case TAB_RESET_PW:
    return (<ResetPW />)
}

```



```

    } else {
      dispatch(setTabRegistration())
    }
  } />)
case TAB_LOGIN:
  return (
    <div className='cover c-login'>
      <LoginTab setTab={() => dispatch(setTabRegistration())}/>
    </div>
  )
case TAB_REGISTRATION:
  return (
    <div className='cover c-login'>
      <RegistrationTab />
    </div>
  )
case TAB_RESET_PW:
  return (
    <div className='cover c-login'>
      <ResetPW/>
    </div>
  )
default:
  return (
    <Lending setTab={() => {
      if (isLogin) {
        history.push('/directions')
      } else {
        dispatch(setTabRegistration())
      }
    }}/>
  )
}
}

```

// lending/index.js

```

import React from 'react';
import IconArrow from '../..../images/arrow.png';
import IconIsleOne from '../..../images/Isle1.svg';

```

```

import IconIsleTwo from '../..../images/Isle2.svg';
import IconIsleThree from '../..../images/Isle3.svg';

import IconIsleWithNumberOne from '../..../images/isleWithNumberOne.svg';
import IconIsleWithNumberTwo from '../..../images/isleWithNumberTwo.svg';
import IconIsleWithNumberThree from '../..../images/isleWithNumberThree.svg';
import IconIsleWithNumberFour from '../..../images/isleWithNumberFour.svg';
import IconIsleWithNumberFive from '../..../images/isleWithNumberFive.svg';

import IconGreenArrow from '../..../images/greenArrow.svg';
import IconHeart from '../..../images/heart.svg';
import IconHand from '../..../images/hand.svg';
import IconSmiley from '../..../images/smiley.svg';
import IconPuzzle from '../..../images/puzzle.svg';

import IconSlug from '../..../images/slug.png';
import IconParty from '../..../images/party.svg';
import IconView from '../..../images/view.svg';
import IconCommunity from '../..../images/community.svg';
import IconFine from '../..../images/icon-fine.svg';
import IconApply from '../..../images/icon-apply.svg';
import IconMeet from '../..../images/icon-meet.svg';
import IconChange from '../..../images/icon-change.svg';

import {Footer} from '../..../components'

import './index.css';
// import style from './Lending.module.css';

import Fade from 'react-reveal/Fade';

const Lending = ({ setTab = () => {} }) => {
  return (
    <div className='lending-container'>
      <div className='lending-greeting' >
        <div className='lending-greeting__titles'>
          <Fade bottom>
            <h6 className='lending-greeting__title lending-greeting__title--top'>
              ЛАКАБО ПРОСИМО
            </h6>
          </Fade>

          <img className='lending-greeting__arrow' src={IconArrow} />

          <Fade bottom>

```

```

    <h6 className='lending-greeting__title lending-greeting__title--middle'>
      У CBIT
    </h6>
  </Fade>

  <Fade bottom>
    <h6 className='lending-greeting__title lending-greeting__title--bottom'>
      ВОЛОНТЕРСТВА
    </h6>
  </Fade>
</div>

  <button className='lending-greeting__button-begin' onClick={setTab}
style={{
  cursor: 'pointer'
}}>ПОЗПОЧАТИ!</button>
</div>

<div className='lending-rules'>
  <h6 className='lending-rules__title'>
    ПРАВИЛА ГРИ
  </h6>

  <div className='lending-rules__section'>
    <div className='lending-rules__item lending-rules__item--first'>
      <img className='lending-rules__item-img' src={IconIsleWithNumberOne}
/>

      <div className='lending-rules__item-wrap'>
        <h4 className='lending-rules__item-title'>РЕЄСТРУЙСЯ</h4>
        <p className='lending-rules__item-description'>Розкажи про себе і
потрапляй у світ волонтерства</p>
      </div>
    </div>

    <div className='lending-rules__item lending-rules__item--second'>
      <div className='lending-rules__item-wrap'>
        <h4 className='lending-rules__item-title'>ОБИРАЙ МІСІЮ</h4>
        <p className='lending-rules__item-description'>Місія - це завдання від
організації. Ти можеш обрати між багатьма варіантами (цікавішу чи
кориснішу). Весь перелік у розділі “МАПА”. Давай діяти, чи ти й досі читаєш
ці правила?</p>
      </div>
    </div>
  </div>

```

```

    <img className='lending-rules__item-img' src={IconIsleWithNumberTwo}
  />
</div>

<div className='lending-rules__item lending-rules__item--third'>
  <img className='lending-rules__item-img' src={IconIsleWithNumberThree}
/>

  <div className='lending-rules__item-wrap'>
    <h4 className='lending-rules__item-title'>ЗДІЙСНЮЙ МІСІЮ</h4>
    <p className='lending-rules__item-description'>Ти ж не думав, що ми
залишимо тебе рятувати світ наодинці? Здійснити місію і вирішити всі питання
тобі допоможе координатор на місці волонтерства</p>
  </div>
</div>

<div className='lending-rules__item lending-rules__item--fourth'>
  <div className='lending-rules__item-wrap'>
    <h4 className='lending-rules__item-title'>ОТРИМАЙ
ВИНАГОРОДУ</h4>
    <p className='lending-rules__item-description'>Після виконання
завдання - отримай винагороду! Твої чесно зароблені бали підтверджує
координатор, а всі виконані місії зберігаються в профілі!</p>
  </div>

  <img className='lending-rules__item-img' src={IconIsleWithNumberFour}
/>
</div>

<div className='lending-rules__item lending-rules__item--fifth'>
  <div className='lending-rules__item-wrap'>
    <h4 className='lending-rules__item-title'>ПУХАЙСЯ ДАЛІ</h4>
    <p className='lending-rules__item-description'>Рятуй світ, накопичуй
бали та отримуй круті плюшки</p>
  </div>

  <img className='lending-rules__item-img' src={IconIsleWithNumberFive}
/>
</div>
</div>
</div>

<div className='lending-advantages'>
  <div className='lending-advantages__section'>
    <h6 className='lending-advantages__title'>ПЛЮШКИ</h6>

```



```

<div className='lending-advantages__section-content'>
  <div className='lending-advantages__section-column'>
    <div className='lending-advantages__item'>
      <img className='lending-advantages__item-img' src={IconFine} />

      <div className='lending-advantages__item-wrap'>
        <h4 className='lending-advantages__item-title'>ЗРУЧНІСТЬ</h4>
        <p className='lending-advantages__item-description'>ВЛАСНИЙ
ПРОФІЛЬ ВОЛОНТЕРСТВА ЗАМІСТЬ ДОВГИХ ГУГЛ ФОРМ</p>
      </div>
    </div>

    <div className='lending-advantages__item'>
      <img className='lending-advantages__item-img' src={IconApply} />

      <div className='lending-advantages__item-wrap'>
        <h4 className='lending-advantages__item-title'>КОРИСТЬ</h4>
        <p className='lending-advantages__item-description'>БЕЗЦІННИЙ
ДОСВІД ДЛЯ ТВОГО РЕЗЮМЕ</p>
      </div>
    </div>

    <div className='lending-advantages__item'>
      <img className='lending-advantages__item-img' src={IconMeet} />

      <div className='lending-advantages__item-wrap'>
        <h4 className='lending-advantages__item-title'>ЗНАЙОМСТВА</h4>
        <p className='lending-advantages__item-description'>ЗУСТРІЧІ З
ПРОАКТИВНИМИ ТА УСПІШНИМИ ЛЮДЬМИ</p>
      </div>
    </div>

    <div className='lending-advantages__item'>
      <img className='lending-advantages__item-img' src={IconChange} />

      <div className='lending-advantages__item-wrap'>
        <h4 className='lending-advantages__item-title'>ЗМІНИ</h4>
        <p className='lending-advantages__item-description'>ТВІЙ
ОСОБИСТИЙ ВНЕСОК У КРАЩЕ МАЙБУТНЄ</p>
      </div>
    </div>
  </div>
</div>

<div className='lending-advantages__section-column'>

```

```

<div className='lending-advantages__item'>
  <img className='lending-advantages__item-img' src={IconParty} />

  <div className='lending-advantages__item-wrap'>
    <h4 className='lending-advantages__item-title'>ДВІЖ</h4>
    <p className='lending-advantages__item-
description'>ВОЛОНТЕРСТВО У КРУТИХ ОРГАНІЗАЦІЯХ ТА НА
ДРАЙВОВИХ ПОДІЯХ</p>
  </div>
</div>

<div className='lending-advantages__item'>
  <img className='lending-advantages__item-img' src={IconView} />

  <div className='lending-advantages__item-wrap'>
    <h4 className='lending-advantages__item-title'>ІМІДЖ</h4>
    <p className='lending-advantages__item-description'>ТВОЯ
УСПІШНА ВОЛОНТЕРСЬКА РЕПУТАЦІЯ</p>
  </div>
</div>

<div className='lending-advantages__item'>
  <img className='lending-advantages__item-img' src={IconCommunity}
/>

  <div className='lending-advantages__item-wrap'>
    <h4 className='lending-advantages__item-title'>ЗГУРТУВАННЯ</h4>
    <p className='lending-advantages__item-description'>СПІЛЬНОТА
ОДНОДУМЦІВ, ЩО ВМІЮТЬ РАЗОМ ВЕСЕЛИТИСЯ</p>
  </div>
</div>
</div>
</div>
</div>
</div>
</div>

<div className='lending-places'>
  <div className='lending-places__section'>
    <div className='lending-places__tasks'>
      <div className='lending-places__tasks-motivation'>
        <h4 className='lending-places__tasks-motivation-title'>
          ЗГАДАЙ ДИТИНСТВО
        </h4>
        <p className='lending-places__tasks-motivation-subtitle'>
          ВОЛОНТЕРЮ ДЛЯ ДІТЕЙ

```

```

    </p>
  </div>

  <div className='lending-places__tasks-item lending-places__tasks-item--
first'>
    <h4 className='lending-places__tasks-item-title'>ТЕАТРАЛІЗАЦІЯ
КАЗКИ «РУКАВИЧКА»</h4>
    <img className='lending-places__tasks-item-img' src={IconIsleOne} />
  </div>

  <div className='lending-places__tasks-item lending-places__tasks-item--
second'>
    <h4 className='lending-places__tasks-item-title'>ВІЗИТ У ДИТЯЧИЙ
БУДИНОК</h4>
    <img className='lending-places__tasks-item-img' src={IconIsleThree} />
  </div>

  <div className='lending-places__tasks-item lending-places__tasks-item--
third'>
    <h4 className='lending-places__tasks-item-title'>ДОПОМОГА З
ПІДГОТОВКОЮ ДО ЗНО</h4>
    <img className='lending-places__tasks-item-img' src={IconIsleTwo} />
  </div>

  <div className='lending-places__tasks-item lending-places__tasks-item--
fourth'>
    <h4 className='lending-places__tasks-item-title'>ІНТЕРАКТИВНЕ
НАВЧАННЯ ДЛЯ ДІТЕЙ З ІНВАЛІДНІСТЮ</h4>
    <img className='lending-places__tasks-item-img' src={IconIsleTwo} />
  </div>

  <div className='lending-places__tasks-arrows'></div>
</div>

<div className='lending-places__article'>
  <h4 className='lending-places__article-title'>
    МІСЦЯ ВОЛОНТЕРСТВА
  </h4>
  <p className='lending-places__article-description'>
    Завітай на будь-яку планету - це світ волонтерств за одним напрямком.
    Згодом вирушай на острівце, який подобається найбільше -
    там знаходиться твоя місія. Залишилось всього лиш натиснути “Хочу!”.
    Усі деталі можна перевірити у своєму профілі.
  </p>
  <p className='lending-places__article-description'>

```



```

    Пам'ятай: твоє основне завдання - виконати місію!
  </p>
  <p className='landing-places__article-description bg-pink'>
    Використовуй стрілки для того, аби переглянути усі планети
    волонтерства у нашому світі
  </p>
</div>
<div className='landing-places__button'>
  <p className='landing-places__button-title'>То як, ти з нами?</p>
  <button
    className='landing-places__button-item'
    onClick={setTab}
    style={{
      cursor: 'pointer'
    }}
  >РОЗПОЧАТИ!</button>
</div>
</div>

<div className='landing-aboutUs'>
  <div className='landing-aboutUs__item'>
    <h3 className='landing-aboutUs__item-title'>
      НАША МІСІЯ
    </h3>
    <p className='landing-aboutUs__item-description'>
      Зробити волонтерство популярним, веселим і ефективним сьогодні у
      Львові, щоб воно стало корисним і впливовим для майбутнього України.
    </p>
  </div>

  <div className='landing-aboutUs__item'>
    <h3 className='landing-aboutUs__item-title'>
      НАША ВІЗІЯ
    </h3>
    <p className='landing-aboutUs__item-description'>
      Ми бачимо волонтерство, що стає можливим завдяки потужному
      інструменту “ВолонтерЮ”, як спосіб розвиватися, отримувати задоволення і
      впливати.
    </p>
  </div>
</div>

<div className='landing-values'>
  <h3 className='landing-values__title'>

```


НАШІ ЦІННОСТІ

```

</h3>

<div className='lending-values__container'>
  <div className='lending-values__item'>
    <img className='lending-values__item-img' src={IconGreenArrow} />
    <h3 className='lending-values__item-title'>КОРИСТЬ</h3>
    <p className='lending-values__item-description'>
      Ми віримо, що особистісне зростання кожного студента-волонтера - це
      внесок у розвиток України
    </p>
  </div>

  <div className='lending-values__item'>
    <img className='lending-values__item-img' src={IconHeart} />
    <h3 className='lending-values__item-title'>ДОБРОЧИННІСТЬ</h3>
    <p className='lending-values__item-description'>
      Ми вважаємо, що намір творити добро є невід'ємною частиною
      мотивації волонтера і основним фактором позитивного впливу на суспільство
    </p>
  </div>

  <div className='lending-values__item'>
    <img className='lending-values__item-img' src={IconHand} />
    <h3 className='lending-values__item-title'>ПРОАКТИВНІСТЬ</h3>
    <p className='lending-values__item-description'>
      Ми впевнені, що у грі народжується ініціативне та відповідальне
      ставлення до всього оточуючого
    </p>
  </div>

  <div className='lending-values__item'>
    <img className='lending-values__item-img' src={IconSmiley} />
    <h3 className='lending-values__item-title'>ЗАДОВОЛЕННЯ</h3>
    <p className='lending-values__item-description'>
      Для нас дуже важливо, щоб кожен учасник отримував задоволення від
      того, що волонтерить. Адже процес є не менш важливим, ніж результат
    </p>
  </div>

  <div className='lending-values__item'>
    <img className='lending-values__item-img' src={IconPuzzle} />
    <h3 className='lending-values__item-title'>СПІВПРАЦЯ</h3>
    <p className='lending-values__item-description'>
      Ми переконані, що можемо змінюватися і змінювати світ на краще
    </p>
  </div>

```

тільки працюючи разом

```

    </p>
  </div>
</div>
</div>

<div className='lending-questions'>
  <img className='lending-questions__img' src={IconSlug} />

  <div className='lending-questions__form'>
    <textarea className='lending-questions__form-textarea'
placeholder='Напиши своє питання' />
    <input className='lending-questions__form-input' placeholder='Введи свою
пошту' />
    <button className='lending-questions__form-
button'>НАДІСЛАТИ</button>
  </div>
</div>
</div>
);
};
export default Lending;

```