**ЗМІСТ**

[ВСТУП 6](#_Toc515183539)

[1. ЗАДАЧА РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ ДІЙСНОЇ МІНІМАЛЬНОЇ ПОВЕРХНІ НА ОСНОВІ ІЗОТРОПНОГО В-СПЛАЙНУ 7](#_Toc515183540)

[2. ЗАГАЛЬНИЙ АНАЛІЗ ПРОБЛЕМ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОГО ДОДАТКУ 8](#_Toc515183541)

[3. ОБЧИСЛЮВАЛЬНІ МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ 16](#_Toc515183542)

[3.1. Середовище розробки 17](#_Toc515183543)

[3.2. Технології, використані при розробці програмного засобу 18](#_Toc515183544)

[4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 30](#_Toc515183545)

[ВИСНОВКИ 42](#_Toc515183546)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 54](#_Toc515183547)

**ВСТУП**

Серед сучасних програмних засобів важко знайти такі, що не використовують клієнт архітектуру для забезпечення доступу до неї. Важливим аспектом побудови модулю, що виконує функції взаємодії з побудовою площин є створення чіткого зрозумілого інтерфейсу з можливістю надання необхідної інформації про стан сплайну та інформування інших частин системи та користувача про їх успішну чи неуспішну обробку.

Повноцінну систему бодування площин неможливо створити без веб-інтерфейсу, адже необхідно забезпечити користувачам доступ за допомогою різноманітних пристроїв із різноманітними операційними системами.

Розв’язком перелічених проблем є використання мови програмування JS як мови програмування для побудови клієнт частини, оскільки на даний момент ця мова програмування є найбільш популярною для таких видів задач і для неї існує багато перевірених часом рішень. Також будуть вікористані біблиотеки Math.js- для складних арифметичних рішень та Plotly.js- для коректного відображення площин. Отже головна задача – це побудува плоскої сітки на основі напрямної, що задається сегментом ізотропної кривої періодичного В-сплайну з нормалізованою параметризацією.

# **ЗАДАЧА РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ ДІЙСНОЇ МІНІМАЛЬНОЇ ПОВЕРХНІ НА ОСНОВІ ІЗОТРОПНОГО В-СПЛАЙНУ**

Метою розробки є реалізація веб-орієнтованого програмного засобу, що надає можливость будування мінімальної дісної поверхні на основі ізотропного В-сплайну .

Призначенням даного програмного засобу є надання чіткого зрозумілого інтерфейсу для побудові площин та усіх необхідних для цього можливостей. Програмний код містить ряд класів, кожен з яких відповідає за певну частину однієї і тої самої задачі, наприклад, при створенні площини системи або виконанні запиту на відображення/редагування площини взаємодію з віддаленою базою даних виконує модельний клас Plotly.

Необхідними можливостями, які має забезпечувати модуль, є:

* можливість перегляду існуючої площини;
* можливість видалення площини
* можливість редагування існуючої площини;

# **ЗАГАЛЬНИЙ АНАЛІЗ ПРОБЛЕМ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОГО ДОДАТКУ**

Згадайте, що ступінь кривої Безьє визначається кількістю задають точок, причому всі вони впливають на форму всієї кривої. Ці особливості кривих Безьє створюють певні незручності. По-перше, при апроксимації кривої складної форми за допомогою кривої Безьє неминуче використовується безліч задають точок, в результаті чого виходить крива високого порядку. Така крива може осциллировать, але вона до того ж створює велику обчислювальну навантаження на комп'ютер. Чому б у такому разі не спробувати уявити ту ж вихідну криву безліччю кривих Безьє нижчих порядків, щоб уникнути цих неприємностей? Проблема в тому, що з'єднання кривих із забезпеченням безперервності похідних потрібних порядків виявляється досить складною процедурою.

По-друге, трудомістким виявляється локальна зміна форми кривої. Здається природним перемістити задають точки поблизу змінюваного ділянки кривої, і дійсно, це призводить до модифікації потрібної області, але разом з нею змінюється вся крива цілком. Ця особливість називається властивістю глобальності змін (global modification property). Глобальність змін небажана при створенні кривих заданої форми, оскільки криві завжди створюються або проектуються шляхом безперервної модифікації грубої форми початкового наближення. У системах автоматизованого проектування бажано наявність прямо протилежного змісту - локальності змін (local modification property).

Описані недоліки кривих Безьє пов'язані з вибором функцій сполучення. Таким чином, нам потрібно вибрати новий набір функцій сполучення, що володіють певними властивостями. По-перше, в визначення нової функції сполучення не повинно входити число точок п, на відміну від функції Вin (і). Ступінь функції сполучення, а значить, і ступінь кривої, повинні бути незалежні від числа задають точок п. По-друге, всі функції сполучення повинні бути відмінні від нуля тільки на обмежених подмножествах значень параметра, причому для кожної функції таке підмножина має бути унікальним. У цьому випадку форма сегмента буде визначатися тільки тими задають точками, які враховуються функціями сполучення, що мають ненульові значення на даному сегменті.

У 1972 р Кокс запропонували використовувати функції Nik (u), що визначаються рекурсивно. Крива, яка будується таким чином, називається В-сплайном (B-spline).

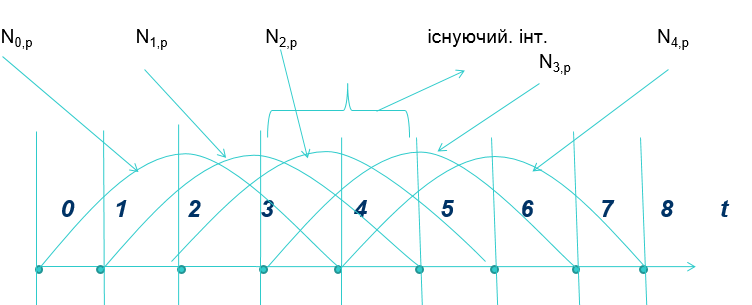
Значення ti називаються вузловими - вони обмежують відрізки значень параметра, всередині яких функції сполучення мають ненульові значення 1. У формулі невизначеність 0/0 вважається рівною нулю. Як випливає з цього рівняння, для визначення п + 1 функцій сполучення необхідно задати п + k + 1 вузлових значень від t0 до tn + k. Різні методи завдання вузлових значень дозволяють отримати різні функції сполучення і, відповідно, різні криві. Нижче ми розповімо, як це робиться. Зверніть увагу, що з фор- мули випливає, що одночасний зсув всіх вузлових значень на одне і те ж число не призводить до зміни форми кривої. При цьому відбувається лише зміна діапазону значень параметра для рівняння .

Перевіримо, чи задовольняють функції сполучення, задані рівняннями і , вимогам, викладеним на початку розділу. З рівняння випливає, що ступінь Nik (u) на одиницю вище, ніж у Nik-1 (u) і Ni + 1, k-1 (u). Слідчий- але, Ni2 (u) має ступінь 1, оскільки Nj 1 (u) - константа, а Ni3 (u) з тієї ж причини має ступінь 2. Продовжуючи в тому ж дусі, можна прийти до висновку, що функція Nik (u ) має ступінь k - 1. Таким чином, ступінь В-сплайна визначається не числом задають точок, а значенням k, яке повинно бути на одиницю більше бажаного значення ступеня кривої. Значення k називається порядком В-сплайна (order of the B-spline curve).

Щоб показати, що будь-яку ділянку В-сплайна визначається лише кінцевим числом задають точок, розглянемо відрізок кривої, відповідний значенням параметра з відрізка [ti, ti + 1]. Впливати на цей сегмент будуть ті що задають точки, функції сполучення яких (близько к) відмінні від нуля на відрізку [ti, ti + 1]. Серед функцій першого порядку від нуля відмінна тільки Ni1 (u). Підставивши Ni1 (u) в праву частину формули , ми отримаємо ненульові на даному відрізку функції Ni2 (u) і Ni - 1.2 (u). Перша функція виходить підстановкою Ni1 (і) в перший доданок, а друга функція - підстановкою у другий доданок. Потім по ненульовим функцій Ni2 (u) і Ni - 1.2 (u) ми отримуємо функції третього порядку і т. Д., Поки не будуть отримані функції порядку k.

Базисні функції періодичного В-сплайна є паралельним переносом один одного і впливу кожної функції поширюються на k-му інтервалі. Для цілих вузлових значень всі базисні функції на одиничного інтервалі 0 <t <1 мають один і той же вид.

Поширення ненульових значень ілюструє положення елементарних В-сплайнів на векторі параметризації n = 4, р = 3, m = 8



Нехай задана певна ізотропна плоска крива . Моделювання будь-яких ізотропних кривих будемо проводити за допомогою завдання дійсних характеристик точок та визначення уявних: . Для побудови сітки використаємо ізотропну криву як напрямну криву. Для цього підставимо замість параметра t певну комплексну змінну *u+iv*: *t=u+iv*.

Оберемо в якості ізотропної кривої періодичний В-сплайн з нормалізованою параметризацією.

Відомо, що періодичний В-сплайн з нормалізованою параметризацією являє собою совокупність сегментів, базісні функції яких приведені до інтервалу параметру  та є однаковими.

|  |  |
| --- | --- |
| , |  |

де j- кількість сегментів кривої В-сплайну,

, , , - реперні точки характеристичного чотирикутника.

Типовий сегмент кривої періодичного В-сплайну четвертого порядку з нормалізованою параметризацією має вигляд.

|  |  |
| --- | --- |
| . |  |

Будемо задавати координати точок характеристичного чотирикутникау комплексному вигляді:

|  |  |
| --- | --- |
| . | (3) |

Для побудови періодичного В-сплайну відокремлюється дійсна частини [6]. При такому підході на формування кривої не впливає уявна складова комплексних складових.Визначимо умови ізотропності для обраного В-сплайну.

Розглянемо моделювання просторової ізотропної модифікованої кривої періодичного В-сплайну четвертого порядку на основі рівняння (3).

Для цього візьмемо квадрат виразу (3) та підставимо в умову ізотропності кривих [6]. Будемо мати:

|  |  |
| --- | --- |
|  |  |

Умова буде виконуватись та не залежати від значення параметра, якщо коефіцієнти при всіх степенях  дорівнюють 0. Тобто одержимо рівняння:

|  |  |
| --- | --- |
|  |  |

Вираз визначає умови ізотропності просторового модифікованого сегменту кривої періодичного нормалізованого B-сплайну четвертого порядку

Розглянемо моделювання плоского ізотропного модифікованого сегменту кривої періодичного нормалізованого B-сплайну четвертого порядку. З урахуванням виразів ординати реперних точок будуть визначатися наступним чином:

|  |
| --- |
|  |

або після спрощення:

|  |
| --- |
|  |

Виділимо окремо дійсну  та уявну  частини та будемо задавати на площині плоску дійсну криву. У цьому випадку кількість умов збільшиться в два рази.Для визначення всіх координат необхідно додати ще дві умови, а саме уявні частини вектору . В результаті одержимо:

|  |  |
| --- | --- |
|  | (  (8) |

Побудуємо плоску сітку на основі ізотропного періодичного нормалізованого B-сплайну четвертого порядку з конформною заміною параметра. Для цього підставимо в :

|  |  |
| --- | --- |
|  | (9) |

Відокремимо дійсну частину від одержаної функції (9). Визначимо внутрішню геометрію побудованої сітки. Для цього розрахуємо коефіцієнти першої квадратичної форми, які дадуть змогу оцінити довжини сегментів кривих, кути між кривими та площі областей на сітці. Для цього візьмемо часткові похідні від , .

Порівняємо одержані похідні:

|  |  |
| --- | --- |
| ,  ,  . | (10) |

Одержані рівняння підставимо у вирази для першої квадратичної форми:

|  |  |
| --- | --- |
| ,  , | (11) |

З рівняння (10) видно, що , а .

Значення  означає, що побудована сітка є ортогональною, а  − ізотермічною.

*Приклад.* Побудуємо ортоганальну та ізотермічну плоску сітку. Для побудови сітки використаємо ізотропну криву сегменту періодичного нормалізованого B-сплайну четвертого порядку. В-сплайн задається вершинами , де перші і останні три вершини – кратні, та складається з п’яти сегментів. Задано такі значення вершин:

,, ,, , , ,, , .

Розрахуємо значення уявних частин на основі виразу :



Сегменти кривої ізотропного модифікованого B-сплайну мають наступний вигляд:

|  |  |
| --- | --- |
|  | (12) |

Побудуємо сегмент кривої B-сплайну, та ортогональну ізотермічну плоску сітку. Візьмемо сегментj=2.

Сегмент кривої ізотропного періодичного нормалізованого B-сплайну зображений на рис.1а. За результатом розрахунку виразів (12) побудуємо ізотропний періодичний нормалізований B-сплайн четвертого порядку з кратними вершинами. В-сплайн, що складається з п’яти сегментів , зображено на рис. 2.1.

|  |  |
| --- | --- |
|  |  |
| Рис. 2.1 а) Сегмент кривої ізотропного періодичного нормалізованого B-сплайну четвертого порядку (j=2)  б )Ізотропний періодичний B-сплайн четвертого порядку з нормалізованою параметризицією | |

B2

B0, B1, B2

B3

B4

B5, B6, B7

B3

B4

B5

Сегмент кривої B-сплайну

(j=2), вершини характеристичного чотирикутника - B2,B3,B4,B5.

а)

б)

# **ОБЧИСЛЮВАЛЬНІ МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ**

Основним середовищем розробки було середовище розробки VisualStudio Code, що є одним із найпопулярніших середовищ розробки для JS, на які можна оформити студентську ліцензію.

Програмний засіб був розроблений з використанням можливостей мови JS та принципів MVC-архітектури на серверній частині; мови розмітки гіпертексту HTML, мови каскадних таблиць стилів CSS та мови програмування JS на клієнтській частині.

Мова програмування JS — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. Мова програмування JS є однією з найпоширеніших мов, що використовуються у сфері веб-розробок

Мова програмування JS інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. Це не є перевага з точки зору безпеки, але покраще інтерактивність сторінок.

Мова програмування JS — мова, код якої можна вбудовувати безпосередньо в html-код сторінок, які у свою чергу, будуть коректно оброблені браузером. Обробник JS просто починає виконувати код після відкриваючого тегу і продовжує виконання до того моменту, поки не зустріне закриваючий тег.

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і web. Важливою перевагою JS є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. Продуктивність JS достатня для створення цілком серйозних веб-застосунків.

**3.1. Середовище розробки**

Інтегроване середовище розробки VisualStudio Code являє собою інтелектуальний редактор для CSS, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторингу для JavaScript. Автодоповнення коду в VisualStudio Code підтримує специфікацію JS (сучасні і традиційні проекти), включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів.

Інтегроване середовище розробки VisualStudio Code було розроблено на основі популярної платформи VisualStudio, написаної на C++. Користувачі можуть розширити функціональність середовища розробки за рахунок встановлення плаґінів, розроблених для платформи VisualStudio, або написавши власні.

Він підтримує ряд мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію по коду, підтримку Git та інші можливості. Багато можливості Visual Studio Code не доступні через графічний інтерфейс, найчастіше вони використовуються через палітру команд або JSON файли (наприклад, призначені для користувача настройки). Палітра команд представляє собою подобу командного рядка, яка викликається поєднанням клавіш.

Visual Studio так само дозволяє замінювати кодову сторінку при збереженні документа, символи перекладу рядка і мову програмування поточного документа.

Visual Studio Code збирає дані про використання (телеметрію) і відправляє їх в Microsoft, але це функція є відключається. Дані можуть передаватися контрольованим філіям Microsoft, дочірнім компаніям і правоохоронним органам відповідно до заяви про конфіденційність.

**3.2. Технології, використані при розробці програмного засобу**

Для розробки програмного засобу було фронтенд-фреймворк Bootstrap (версія 3.3) , бібліотеки Math.js, Plotly.js, MVC-архітектура Vue.js (версія 2.0), та стандарти мови розмітки HTML5 .

Фреймворк Math.js - це велика математична бібліотека для JavaScript та Node.js. Він має гнучкий аналізатор виразок з підтримкою символічного обчислення, поставляється з великим набором вбудованих функцій і констант і пропонує інтегроване рішення для роботи з різними типами даних, такими як цифри, великі числа, складні числа, фракції, одиниці та матриці. Потужний і простий у використанні. Особливості:

1) Підтримує числа, великі числа, складні числа, фракції, одиниці, рядки, масиви та матриці.

2) Сумісність з вбудованою в JavaScript математичною бібліотекою.

3) Містить гнучкий аналізатор виразок.

4) Є символічне обчислення.

5) Поставляється з великим набором вбудованих функцій та констант.

6) Можна також використовувати як додаток командного рядка.

7) Працює на будь-якому движку JavaScript.

8) Легко розширюваний.

9) Відкрите джерело коду.

Цей фреймворк допомогая нам в будуванні комплексних массивів , та операцій над ними.

Ядро math.js - це простір імен математики, що містить всі функції та константи. Є три способи зробити обчислення в math.js:

1)Використовуйте виклики звичайної функції, як math.add (math.sqrt (4), 2).

2)Оцінюючи вирази, такі як math.eval ('sqrt (4) + 2')

3)Об'єктивні операції, такі як math.chain (4) .sqrt (), add (2).

Math.js - велика бібліотека, що містить багато типів даних і функцій. Цілком можливо, що вам потрібна невелика частина бібліотеки. Math.js дозволяє створювати індивідуальний файл індексу, завантажуючи лише ті типи даних і функції, які вам потрібні. Це дає швидший час завантаження та дозволяє об'єднувати тільки використану частину бібліотеки за допомогою таких інструментів, як Webpack або browserify.

Щоб завантажити порожній екземпляр math.js, завантажте mathjs / core. Це ядро містить лише функції імпорту та конфігурації.

Потім використовуйте math.import для завантаження необхідних типів даних та функцій. Перш за все, важливо завантажувати типи даних, а потім - функції завантаження та константи. Ці функції динамічно будуються залежно від наявних типів даних.

Фреймворк Plotly.js пропонує безліч функцій, які допомагають вивчати бібліотеку. Це декларативна бібліотека високого рівня, побудована на вершині d3.js і stack.gl. Нижче наведено список функцій, які зроблять Plotly одним із найкращих бібліотек графіків JavaScript.

Ви можете легко створювати інтерактивні діаграми за допомогою Plotly.js. Будь-яка діаграма, яку ви створюєте в бібліотеці, обладнана такими функціями, як масштабування, масштабування, панорамування, автоматичне масштабування тощо. Ці особливості дуже корисні, якщо ви хочете вивчати діаграми з великою кількістю графічних точок. Усі ці події виставляються в API, тому ви можете створити власний код, щоб виконувати власні дії, коли виникає будь-яке з цих подій.

Висока продуктивність при побудові багатьох моментів робить Plotly.js чудовим вибором кожного разу, коли потрібно охарактеризувати велику кількість даних. Оскільки більшість графіків створюються за допомогою SVG, ви отримуєте пристойну кількість сумісності між браузерами та можливість експортувати якісні зображення будь-якої діаграми. Проте, малювання великої кількості елементів SVG у DOM може негативно вплинути на продуктивність. У бібліотеці використовується stack.gl для створення високопродуктивних 2D та 3D-графіків.

Будь-які 3D-графіки, які ви створюєте, створюються за допомогою WebGL, щоб повністю використовувати всі можливості, які GPU може запропонувати.

Всі діаграми Plotly.js повністю настроюються. Все з кольорів та міток до ліній сітки та легенд можна налаштувати, використовуючи набір атрибутів JSON. Ви дізнаєтеся, як налаштувати різні типи графіків у наступних трьох частинах серії.

Фреймворк Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-застосунків. Це — клієнтський фреймворк, тобто призначений для розробки інтерфейсу користувача, на відміну від інструментів серверної сторони. Фреймворк Bootstrap — інтуїтивно простий і у той же час потужний інтерфейсний фрейморк, що підвищує швидкість і полегшує розробку web-додатків.

Фреймворк Bootstrap був розроблений як фреймворк для забезпечення однаковості внутрішніх інструментів компанії Twitter. До появи Bootstrap у розробці інтерфейсу їх внутрішніх інструментів застосовувалися різні бібліотеки, що призводило до появи суперечностей та ускладнювало супровід.

Реліз із відкритим сирцевим кодом вийшов 19 серпня 2011 року. Нині проект підтримується невеликою групою розробників, а також широкою спільнотою прихильників.

Фреймворк Bootstrap сумісний з останніми версіями браузерів Google Chrome, Firefox, Internet Explorer, Opera і Safari.

Фреймворк Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних .

Наприклад, в імперативному програмуванні вираз, a := b + c означає, що a отримує результат виконання b + c безпосередньо під час обчислення виразу, і потім, якщо значення b і c зміняться, це не впливатиме на вже обчислене значення a.

Проте, в реактивному програмуванні значення a буде автоматично оновлено за новими значеннями, що є протилежністю функційного програмування.

Сучасна програма електронної таблиці є прикладом реактивного програмування. Комірки електронної таблиці можуть мати буквальні значення, або формули типу “=B1+C1”, що обчислюються за значеннями інших комірок. Коли б не змінилося значення іншої комірки, значення формули оновлюється автоматично.

Реактивне програмування спочатку пропонувалося як засіб простого створення інтерфейсів користувача, анімацій у системах реального часу, але стало загальною парадигмою програмування.

Наприклад, у архітектурі Модель-Представлення-Контролер, реактивне програмування дозволяє змінам у моделі автоматично відображатися у виді, і навпаки.

Реактивне програмування може бути чисто статичними, де потоки даних встановлюються статично, або бути динамічним, де потоки даних можуть змінюватися під час виконання програми.

Використання умовних переходів в графі потоку даних може в деякій мірі змусити статичний граф потоку даних виглядати як динамічний, і злегка затушувати відмінність. Проте, чисто динамічне реактивне програмування може використовувати імперативне програмування для реконструкції графа потоку даних.

Обчислення, чи оцінювання у реактивних програмах не обов'язково засновано на тому ж, що й обчислення у програмах, що будуються на стеках. Замість цього, коли якісь дані змінюються, то зміна поширюється на всі дані, які були отримані повністю або частково зі змінених даних. Це поширення зміни може бути досягнуто кількома способами, де, можливо, найприроднішим способом є схема інвалідація/лінива-ревалідація [2].

Одна проблема, притаманна реактивному програмуванню, це те, що більшість обчислень, які у звичайній мові програмування були б зроблені і забуті, мусять лишатися в пам'яті у вигляді структур даних. Це може зробити реактивне програмування дуже вимогливим до об'єму пам'яті. Проте, дослідження, що називається “зниженням”, має потенціал вирішити цю проблему.

З іншого боку, реактивне програмування є формою того, що можна описати як “явний паралелізм”, і, таким чином, може бути корисним для використання потужностей паралельного апаратного забезпечення.

Vue - це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро ​​в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Творцем Vue.js є Evan You, колишній співробітник Google і Meteor Dev Group. Почав він розробляти фреймворк в 2013-му, а в лютому 2014 го відбувся перший публічний реліз. Vue широко використовується серед китайських компаній, наприклад: Alibaba, Baidu, Xiaomi, Sina Weibo і ін. Він входить в ядро ​​Laravel і PageKit. Нещодавно вільна система управління репозиторіями GitLab теж перейшла на Vue.js.

В кінці вересня 2016- го вийшов в реліз Vue.js 2.0, ще крутіше і з упором на продуктивність - тепер використовується віртуальний DOM, підтримується серверний рендеринг, можливість використовувати JSX і т.д. Хоча зараз він підтримується тільки співтовариством, він тримається гідно навіть на рівні продуктів таких гігантів, як Google і Facebook (Angular2 і React 15), і поступово наздоганяє їх по популярності.

Технічно, Vue.js визначена як ViewModel шар шаблону MVVM. Вона з'єднує модель і подання до двостороннє зв'язування даних. Поточні DOM-зміни і форматований вивід абстрагуються в Директивах і фільтрів.Мета надати переваги швидких зв'язувань даних і складних уявлень компонентів з API як прості, так і зрозумілі. Бібліотека не є повномасштабним фреймворком, вона є всього лише рівнем уявлення. Ви можете використовувати її як окремо, для швидкого прототипування застосувань, або змішувати і поєднувати з іншими бібліотеками для кастомізації користувальницького інтерфейсу. Але чому саме Vue.js? Адже є інші фреймворки такі як Angular чи React які більш популярні, та вже перевіренні часом і також в цих фреймворках багато готових рішень відображення цих графіків. Але у Vue.js є ціла купа люсів порівнянно з іншими фреймворками.

**Є кілька причин використовувати Vue.js замість Angular:**

1. Vue.js більш гнучкий, менш прямолінійний в рішення. Це дозволяє структурувати ваш додаток як ви хочете, а не змушує робити все це як в Angular. Vue є тільки шаром уявлення, тому ви можете використовувати його як ненав'язливу особливість на сторінці додатка, замість повномасштабного фреймворка. Це дає вам більше можливостей для поєднання Vue з іншими бібліотеками, але ви також несете відповідальність за більш архітектурні рішення. Так, наприклад, ядро ​​Vue.js не включає маршрутизацію або ajax-функції з коробки, і, як правило, передбачає, що ви будуєте додаток, використовуючи зовнішній модуль. Це, ймовірно, найбільш важлива відмінність.
2. Vue.js набагато простіше, ніж Angular, як з точки зору API, так з точки зору дизайну. Ви можете вивчити практично всі аспекти Vue.js досить швидко, і відразу почати писати.
3. Vue.js має більш високу продуктивність, тому що не використовує "dirty checking". Angular стає повільнішим, коли використовуються безліч спостерігачів, тому що кожен раз при змінах в області видимості, всім цим спостерігачам необхідно обчислюватися знову. Vue.js не страждає від цього, тому що використовує прозору систему відстеження спостерігачів, тому всі зміни викликаються самостійно, при наявності явних пов'язаних залежності.

**Є кілька причин використовувати Vue.js замість React:**

1. React і Vue.js мають невелику схожість в тому, що обидва підтримують реактивні і компонований уявлення компонентів. Проте внутрішня реалізація в корені відрізняється. React заснований на віртуальному DOM - в пам'яті представлено те, як DOM виглядає насправді. Дані в React в значній мірі є незмінними, і DOM-зміни розраховуються на основі визначення відмінностей. У Vue.js, навпаки, дані змінювані за замовчуванням, і змінюються через події. Замість віртуального DOM, Vue.js працює безпосередньо з DOM як шаблон, зберігаючи посилання на існуючі вузли для зв'язування.
2. Підхід з віртуальним DOM забезпечує функціональний спосіб описати ваше уявлення в будь-який момент, це дуже приємно. Тому що не використовується шаблон спостерігачів і не перемальовується все додаток при кожному оновлення, уявлення щодо визначення гарантує бути синхронізуються з даними, що також відкриває можливості для ізоморфних JavaScript-додатків.
3. Але проблема при використанні React в тому, що ваша логіка і уявлення тісно переплітаються. Для деяких розробників це плюс, але для дизайнерів / розробників гібридів, представляти шаблон наочно набагато простіше в дизайні і CSS. JSX зливаючись з Javascript-логікою ламає ту візуальну модель, яка необхідна для відображення коду в дизайні. Vue.js, на відміну, розплачується своїми легкий директивами, але у вас завжди є візуально представлений шаблон і інкапсульована логіка в директивах і фільтрах.
4. Ще одна проблема React полягає до тому, що всі оновлення делеговані на віртуальний DOM, це трохи складно, коли вам насправді потрібно контролювати DOM самостійно (хоча теоретично можна, слід по суті працювати на противагу бібліотеці, коли це необхідно). Для додатків, яким потрібна складно-синхронізована анімація за часом це може стати дуже дратівливою обмеженням. На цьому фронті Vue.js є більш гнучким.

Тим паче , концепція Vue.js дуже проста , а саме:

* Конструктор
* Компоненти
* Директиви
* Переходи

Робота з Vue.js починається зі створення нового інстанси new Vue. У el у нас елемент, за яким стежить Vue. У template обраний (або прописаний інлайново) елемент, куди Vue буде рендерить. У data зберігається поточний стан інстанси, а метод computed надає нам обчислювані властивості.

У прикладі обчислюється властивість full\_name відстежує first\_name і last\_name як залежності і автоматично синхронізується.

У methods можна виділити наступні кастомниє методи і методи життєвого циклу Vue (рис 3.1):

* beforeCreate - дивиться дані і ініціалізує події
* created - дивиться, чи є el або template. Якщо є, то рендерить в них; якщо немає, то шукає метод render
* beforeMount - створює vm. $ el і замінює їм el
* mounted - елемент отрендеріть

При изменении состояния:

* beforeUpdate — снова рендерит VDOM и сравнивает с реальным DOM-ом, применяет изменения
* updated — изменения отрендерены
* beforeDestroy — полный демонтаж вотчеров, внутренних компонентов и слушателей событий
* destroyed — вызывается, когда выполнение операции останавливается

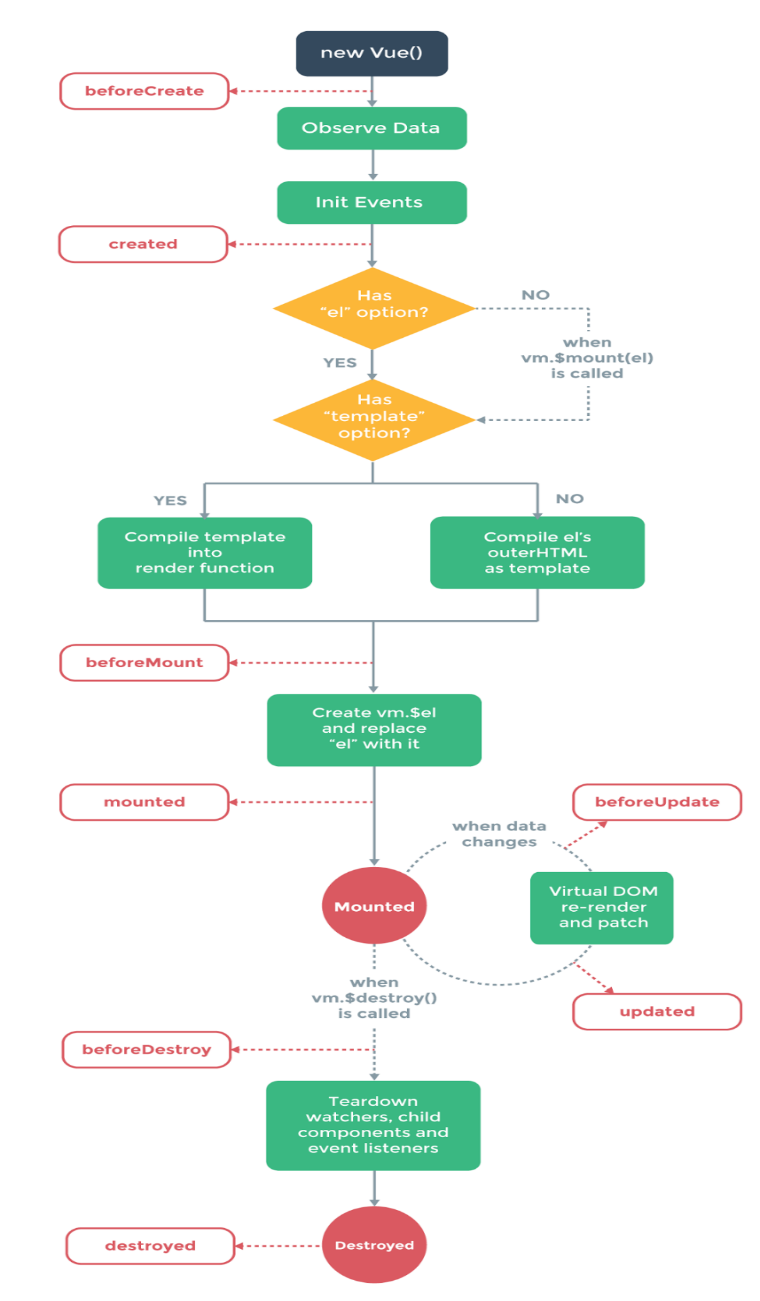


Рис 3.2.1-життевий цикл Vue

Директиви - спеціальні атрибути для додавання елементів html додаткової функціональності. Декілька прикладів директив:

* V-bind - динамічно зв'язується з одним або декількома атрибутами.
* V-cloak - ховає "вусаті" вираження, поки не підтягнулися дані
* V-if - умова для рендера елемента
* V-else - позначає "else блок" для v-if
* V-for - циклічно проходить масив об'єктів
* V-model - пов'язує стан з input елементом
* V-on - пов'язує слухача події з елементом
* V-once - рендерить елемент тільки спочатку і більше не стежить за ним
* V-pre - НЕ компілює елемент і його дочірні елементи
* V-show - перемикає видимість елемента, змінюючи властивість CSS display
* V-text - оновлює textContent елемента

Компоненти допомагають розширити основні html елементи і впровадити переіспользуемий код. По суті, компоненти - повторно використовувані частини UI. На етапі проектування ми розбиваємо наш додаток на незалежні частини і отримуємо деревоподібну структуру компонентів (рис 3.2.2).

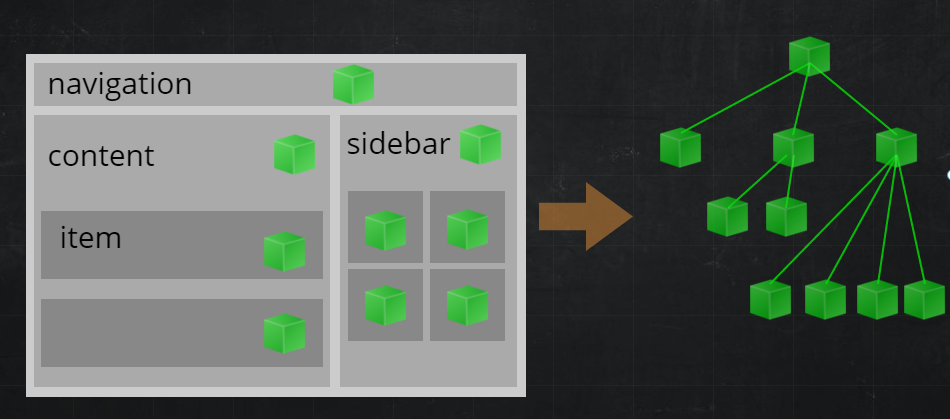


Рис 3.2.2-Архітектура комнонентів

Комунікація між vue-компонентами здійснюється за принципом "Props in, Events out". Тобто від батьківського елемента до дочірньому інформація передається через пропси, а назад - викликаються події (рис 3.2.3).

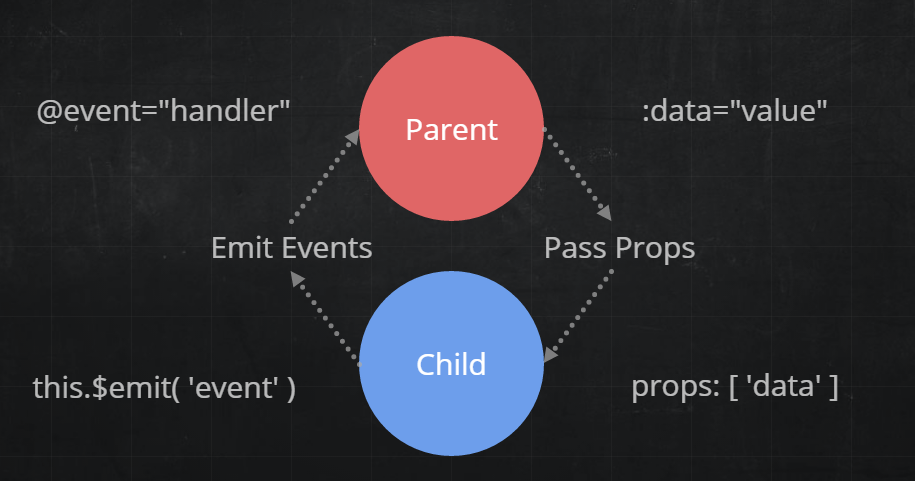


Рис 3.2.3-Комунікація між компонентами

Але іноді може статися так , що нам потрібно змінити інформацію в іншому не зв’язному з цим компоненту. Для цього використовується конструкція **$emit $on**.Отже якщо ми робимо якусь подію , наприклад натискання на кнопку, на подію натискання ми також пишемо **$emit** і назву події . Після цього в компоненті в кому потрібно змінити значення ми повинні підписати цей компонент на цю подію. Підпись робиться завдяки **$on** і назви події що ви написали в іншому компоненті.

Також під Vue.js є так звані однофайловий компоненти. Ми створюємо файл з розширенням .vue і пишемо туди стилі, шаблон і логіку. Причому писати можна на будь-якому зручному вам препроцесорів (SASS, Stylus, PostCSS, Jade, ...) і мовою, компілює в JS (CoffeeScript, TypeScript).

Vue надає різні способи застосування анімаційних ефектів, коли елементи намальовані, оновлені або видалені з DOM. Вони включають в себе інструменти для:

* автоматичного застосування класів для CSS-переходів і анімацій
* інтеграції сторонніх бібліотек для CSS-анімації, таких як Animate.css
* використання JavaScript для маніпуляції DOM-му
* інтеграції сторонніх JavaScript бібліотек для анімацій, таких як Velocity.js

Але одна з найголовніших плюсів Vue є управління станом компонентів , а нзиваєтся він Vuex. Vuex - патерн і бібліотека управління станом для додатків на Vue.js. Він надає централізоване загальний стан для всіх компонентів в додатку і правила, що забезпечують передбачуване зміна стану.

На Vuex зкладається з наступних частин (рис 3.2.4):

* Стан (State), яке є єдиним джерелом даних для компонентів.
* Vue-компоненти (Vue-components), які по суті є лише декларативним відображенням стану.
* Дії (Actions), які відловлюють подія, яка сталася, збирають дані з зовнішніх API і запускають потрібні мутації.
* Мутації (Mutations) - єдина частина, яка може змінювати стан і, отримавши дані від Actions, застосовує їх на стані.

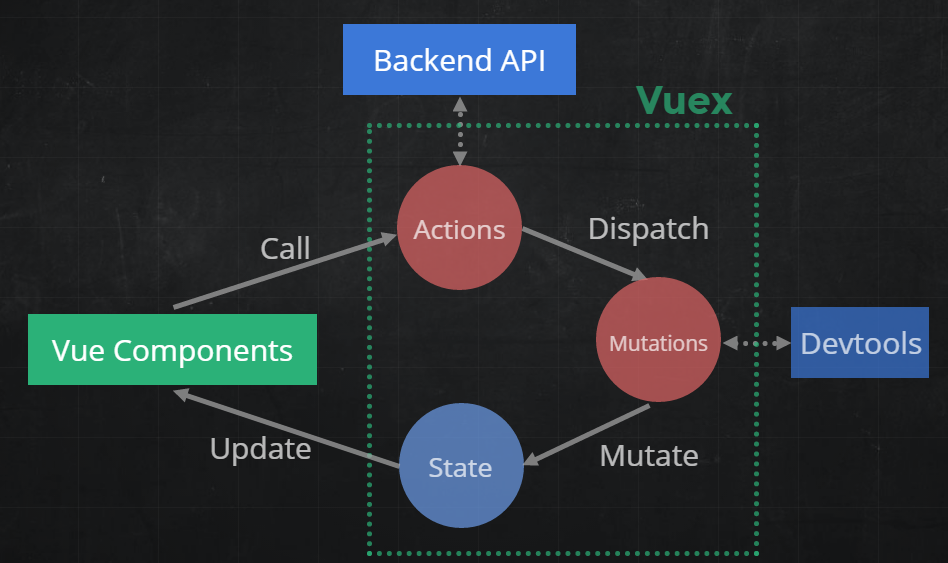


Рис 3.2.4-Склад Vuex

**Отже чому я повинен використовувати Vue.js:**

Vue КРАЩИЙ! Я вважаю, це здорово, що я можу просто підключити його на сторінці і почати працювати без необхідності шаманів з системою складання або CLI. З ним дуже просто почати працювати, навіть якщо ви ніколи не працювали з JavaScript фреймворками. Це ідеальне поєднання зручності і потужності. Розглянемо ще кілька аргументів на його користь:

Тепер він ще менше. Runtime збірка Vue.js 2.0 важить всього лише 16kb, а разом з vue-router і vuex - 26kb (рис 3.2.5).

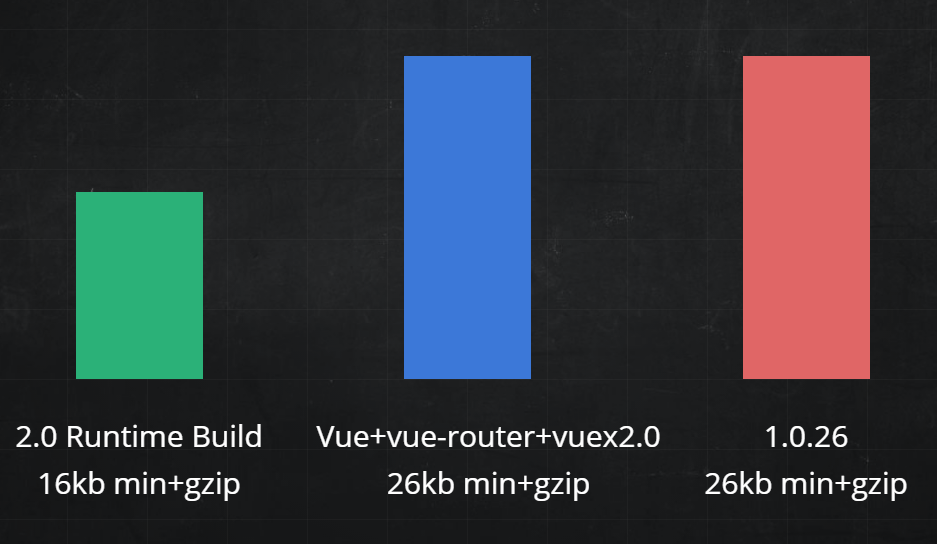


Рис 3.2.5- Навантаженість проекта

Він ще швидше. У Vue.js завжди приділялася велика увага швидкодії. Прошарок, що відповідає за рендеринг, переписана на полегшену реалізацію віртуального DOM-а - Форк Snabbdom. Якщо вибирати з іншими MVC – системами то нова версію Vue.js приблизно на тридцять процентів швидше ніж Angular.Та приблизно на тридцять три процента швидкіша за React. А через те , що ми графічно будуємо площини це дуже ресурсозатратний процесс тому швидкість виконання для нас найголовніше. Виходячи з цих причин я вибрав Vue.js (рис 3.2.6).

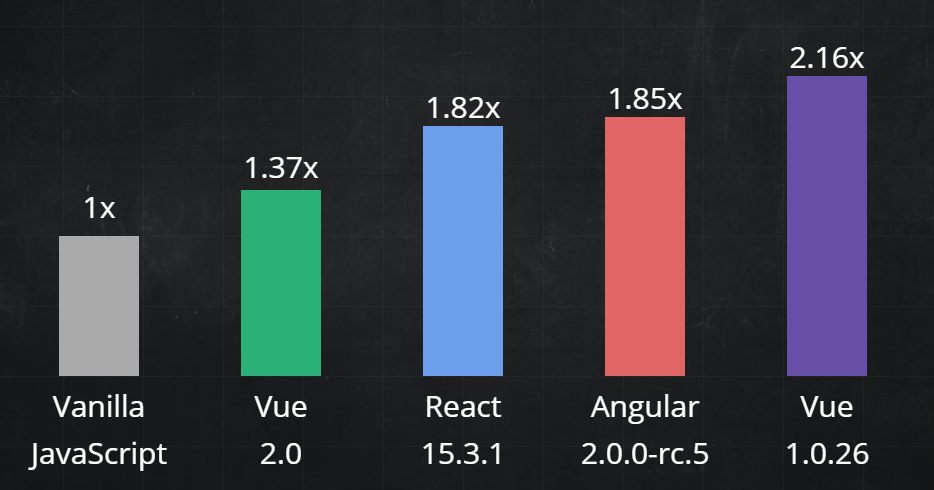


Рис 3.2.6-швидкість Vue.js порівняно з іншими MVC

Vue 2.0 підтримує серверний рендеринг. Для цього є модуль Vue-server-renderer, що підтримує інші інструменти з екосистеми Vue (vue-router і vuex). Тепер набагато простіше створювати ізоморфні додатки.

Коли ми вже розібралися з MVC перейдемо до HTML5. В моєму дипломі використовуэться два нововедення HTML5 , а саме **canvas** та **svg**. Але спочатку розберемо HTML5 більш детально.   
Мова розмітки HTML дотримується правил, які містяться у файлі оголошення типу документа (Document Type Definition, або DTD). DTD є XML-документ, що визначає, які теги, атрибути і їх значення дійсні для конкретного типу HTML. Для кожної версії HTML є свій DTD. DOCTYPE відповідає за коректне відображення веб-сторінки браузером. DOCTYPE визначає не тільки версію HTML (наприклад, html), але і відповідний DTD-файл в Інтернеті.

Canvas- елемент, призначений для створення реєстрового двомірного зображення за допомогою скриптів, зазвичай на мові JavaScript, визвати його потрібно через тег HTML5 <canvas></canvas>. Початок відліку блоку знаходиться зліва зверху. Від нього і будується кожен елемент блоку. Розмір простору координат не обов'язково відображає розмір фактичної інформації, що відображається площі. За замовчуванням його ширина дорівнює трьомстам пикселям, а висота ста п'ятдесяти.

Використовується, як правило, для відтворення графіків для статей і ігрового поля в деяких браузерних іграх. Але також може використовуватися для вбудовування відео в сторінку і створення повноцінного плеєра.

Використовується в WebGL для апаратного прискорення 3D графіки.

**4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ**

Розроблений програмний засіб складається з наступних основних частин c клієнтської частини. Розглянемо її детальніше.

Через те що в цій роботі використовується сучасна MVC модель а саме Vue.js. Спочатку необхідно встановити його та створити проект , але ми будемо використовувати на саме Vue , а один з його фрєймворків Vue-cli. Для цього спочатку глобально встановлюємо Vue-cli (рис 4.1).



Рис 4.1- Глобально встановлюємо vue-cli

Після цього необхідно створити проект с використанням Vue-cli, виглядати

це буде наступним чином Рис 4.2



Рис 4.2 – Створення проекта за допомогою Vue-cli

Після цього в нас буде відкрита структура нашого проекта (рис 4.3) .

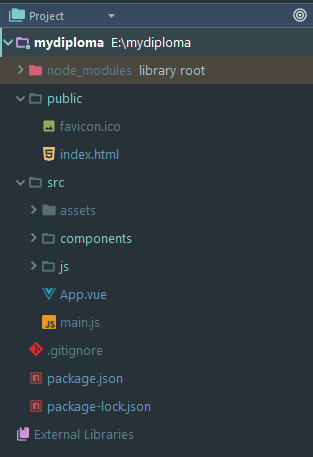


Рис 4.3-Структара проекта Vue-cli

Після цього нам потрібно створити комнонент в якому буде будуватися наші поверхні (рис 4.4).

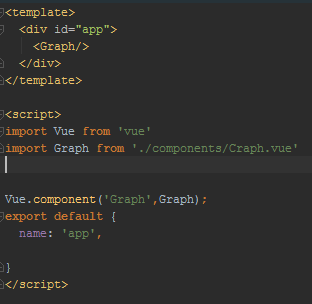


Рис 4.4-створення компонента для побудовання поверхонь

**Template**- це блок компонента в якому буде знаходитися наш HTML код.

**Script**- в цьому блоці буде підключення інших компонентів, передача даних та інше.

Компонент в якому буду відображення наших площин називається Graph. Для його підключення в теги script мы пишемо :

1. **import** Vue **from** 'vue'- тут ми визиваємо глобальний компонент Vue , до якого будуть підєднуватися інші компоненти
2. **import** Graph **from** './components/Craph.vue'- тут мы дамо назву новому компоненту , а саме Graph та вказуємо шлях де цей компонент знаходиться а саме у папці src/components/Craph.vue
3. Vue.component(‘Graph’,Graph)- тут ми під’єднюємо до глобального компонента Vue наш новий компонент Graph і вказуємо його місце .

Далі переходимо до самого компонента Сraph.vue для побудування площин.

Візуальна частина Craph.vue виглядаю ось так (рис 4.5):

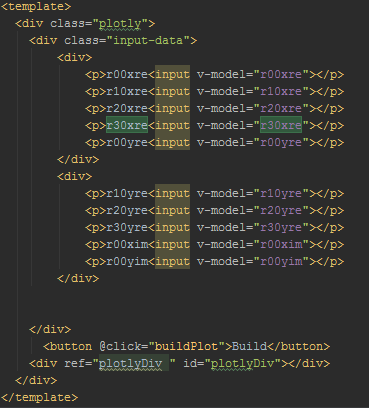


Рис 4.5- HTML розмітка Craph-vue.

Головним компонентам для відображення в нас є елемент div з id =**plotly-div.** Саме він підьєднан до бібліотеки plotly.js. Але також ми маємо десять полів для вводу даних. Тут ви можете побачити таку концепцію , що в нас є v-model. Це означає, що в нашому полях буду знаходиться наша змінна яка вказується в дужках v-model .Спочатку ми об’явимо глобальну зміну math яку будем використовувати для операцій над комплексними числами, вона на пряму буду використовувати із бібліотеки math.js. Виглядати вона буде ось таким чином (рис 4.6):



Рис 4.6-об’ява глобальної змінної math

Після цього через те що я використовую Vue в цьому компоненті в додатковому полі data нам треба позначити ще дві зміні які необхідні нам будут для побудови площин , а саме dataPlot , layout та змінні необхідні для побудови площин. Виглядати це буде наступним чином (рис 4.7)

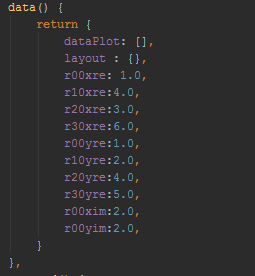


Рис 4.7-об’ява зміних dataPlot та layout

Після цього через того , що ми використовуємо ізотропічний сплайн нам необхідні будуть коплексні числа тому ми одразу зробимо змінну **і** яка буде позначати комлексне значення (рис 4.8)



Рис 4.8 – Позначення комплексного числа і

Далі ми підходимо до коду який саме генерую наші площини на основі изотропного В-сплайну bspline.js.

Спочатку ми повинні будемо об’явити наші масиви комплексних чисел з яких і малюються наші площини. Для початку побудуэмо площину АХ , яка буде відповідати положеню точок на осі Х (рис 4.9).

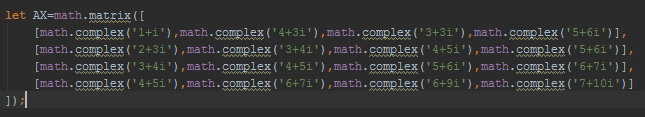


Рис 4.9- комслексна матриця AX

Після цього требо зробити аналогічну матрицю для осі Y и назвемо її AY (рис 4.10)

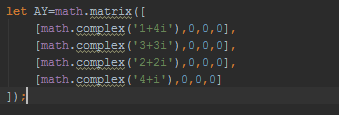


Рис 4.11- комслексна матриця AY

Також к нас є матриця AZ яку мі повинні були використовувати для змін площин на осі Z, але сміщення по осі Z ми не робимо отже вона просто нульова матриця.

Після деяких змін ми створюємо два нових массива які необхідні нам для побудови вершин В-сплайну (рис 4.12).



Рис 4.12-Масив вершин В-сплайну

Також потрібний два массива для побудови ізотропного періодичного B-сплайну четвертого порядку з нормалізованою параметризицією (рис 4.13)

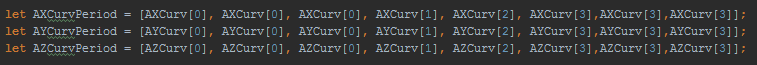


Рис 4.13- масив ізотропного періодичного B-сплайну

Після цього ми можем порахувати координати точок необхідних для , будування цих вершин. Для цього добавимо змінну kkline в яку запишемо усі необхідні координати для побудови за допомогую цикла (рис 4.14) и трьох необхідних формул:

XX := Re((AXCurv[j+1]-AXCurv[j])\*v+AXCurv[j]);

YY := Re((AYCurv[j+1]-AYCurv[j])\*v+AYCurv[j]);

ZZ := Re((AZCurv[j+1]-AZCurv[j])\*v+AZCurv[j])

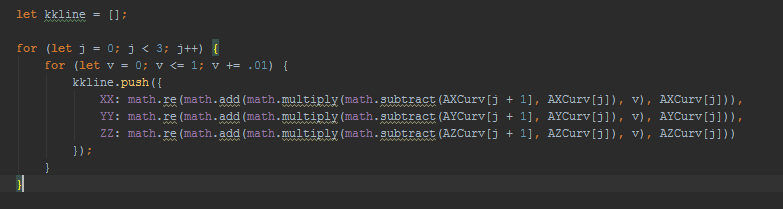


Рис 4.14-Цикл для kkline

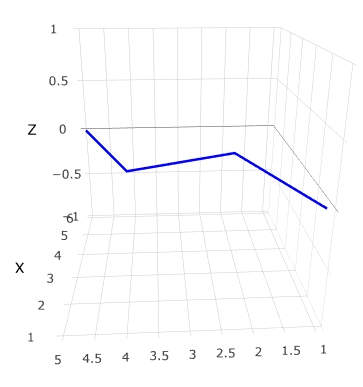
Отже ми отримали масив точок завдяки якому зможемо побудувати лінію на якій знаходяться вершини В-слайну , в нас чотири вершини , а отже лінія буде змінювати напрям чотири рази (рис 4.15). 

Рис 4.15-Приклад лінії вершин В-слайну

Далі ми повині побудувати наш В-сплайн.

Для того, щоб побудувати наш В-сплайн нам треба об’явити масив комплексних чисел , який буде називатися tvector та буде на проміжку від 0.1 до 1 (рис 4.16 ). Також потрібна матриця констант (рис 4.17)



Рис 4.16-tvector

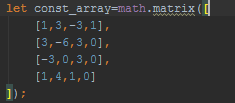


Рис 4.17-матриця констакт

Також необхідно реалізовувати формули для точок площини на X, Y, Z.



Де t - елемент tvector , а для координат Y, Z використовується AYCurvPeriod та AZCurvPeriod. Отже наш ізотропний В-сплайн буде мати вигляд кривої (рис 4.18).

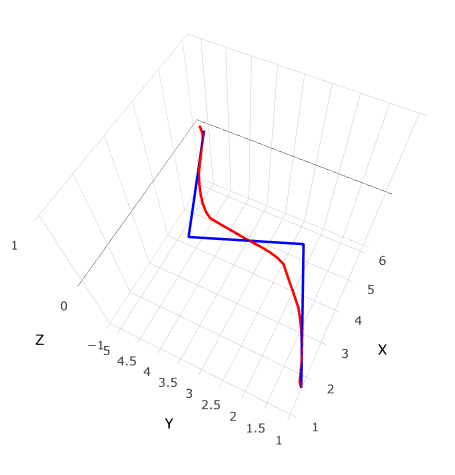


Рис 4.18- Вид В-сплайну четвертого порядку (червона лінія)

Після побудови В-сплайну ми можемо побудовати площини някі знаходяться на цьому сплайні. Для цього треба додати ще декілька змінних, а саме массиви з координатами площин (рис 4.19) та необхідні массиви для їх побудовання (рис 4.20).

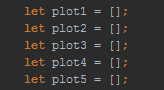


Рис 4.19- Массиви з координатами площин

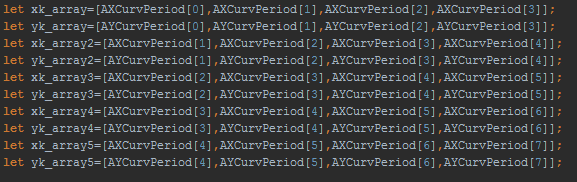


Рис 4.20- необхідні массиви для побудовання площин

А зараз на приклади першої площини ми розбиремо як будується площина plot1 (рис 4.21) и саме через що воно будується. На рисунку 4.20 показано десять массивів які потрібні для цього, отже xk\_array1 та yk\_array1 будуть для plot1, xk\_array2 та yk\_array2 будуть для plot2 і так далі.

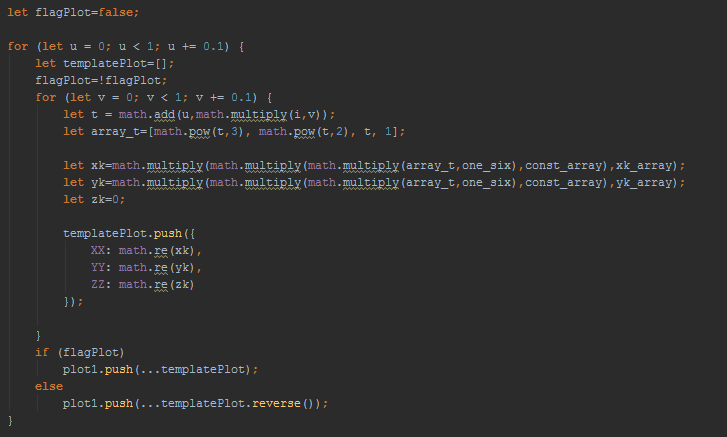


Рис 4.21- Побудова площини plot1

На початку ви бачите змінну **flagPlot**- вона необхідна нам для правильного відображення нашої площини , ця проблема виникла через те що бібліотека Plotly іноді може неправильно відображати нашу площину , а з'єднювати точки площини в неправильному напрямку тому я використовую цю зміну для запису одного цикла массива в зворотньому порядку , щоб відображення будо більше детальне та вірне. Далі ми позначаємо масив **templatePlot**- він необхідний для запису точок , для правильного відображення. На кінці цього малюнку можна побачити саму ту умову де ми використовуємо ці змінні для того, щоб правильно записати данні в наш масив plot1, який будемо використовувати для відображення площин, отже розберемо цю умову більш детально (рис 4.22).

Головним аспектом ціеї перевірки є те що вона максимально оптимальна.

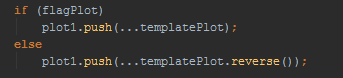


Рис 4.22 – умова для добавлення значення в plot1

Як ви можете побачити , якщо значення **flagPlot** невірне тобто **false** то ми записуємо в **plot1** обернений масив  **templatePlot** і навпаки. Далі ми повинні реалізувати дві формули одна для знаходження координат Х , інша для координат Y.

По коду видно що масив **t** є змінна **array\_t**, а наш массив с числами це наша змінна **const\_array**. Отже на кінці ми отримаємо масив **plot1** в якому буде сто двадцять один елемент (рис 4.23)



Рис 4.23- Масив **plot1**

Якщо ми розкриємо його то зможемо побачити які саме координати має наша перша площина (рис 4.24).

Через те що наші площини не змінюються на площині Z то всі наші Z-координати дорівнюють нулю.

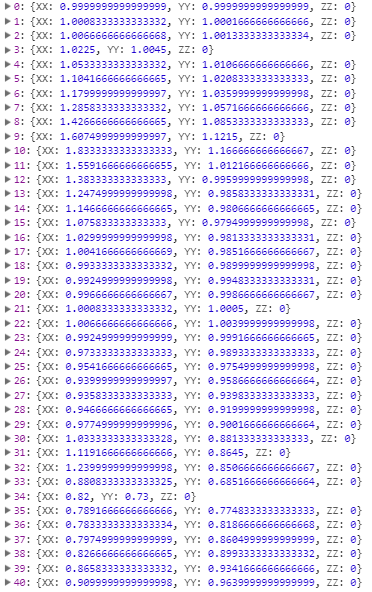


Рис 4.24-розвернутий масив площини

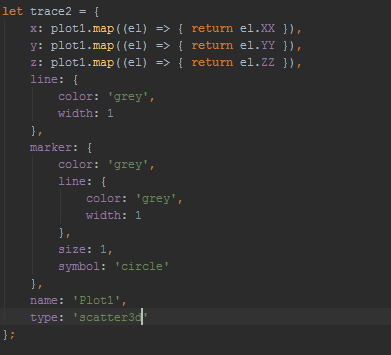
Після побудованого масиву ми нарешті зможемо дійти до його відображення для цього спочатку треба зробити змінну яка буде нашим об’єктом в який ми запишемо правила для відображення, назвемо цю зміну **trace2** (рис 4.25). 

Рис 4.25-Змінна **trace2**

У цьому об'єкті є три обов’язкові змінні для побудови площини це **X,Y,Z**. У ці змінні ми заносимо усі координати змінної **plot1**. Наступна змінна це **line** вона відповідає за те якого коліру та якої товщини буде лінія яка формує нашу площину. Далі іде **marker** ця змінна відповідає за точки по яким будується лінія яуого вони будуть розміру та кольору. Після цього ми задаємо назву нашої площини а також вибираємо тип відображення.

Після цього цю змінну треба вписати до масива тому що така площина буде не одна томі і створюємо масив (рис 4.26).



Рис 4.26-Масив площин

Далі створюємо наш елемент layout, ця змінна буде відповідати за відображення нашої сітки на якії будуть відображатися площини (рис 4.27)

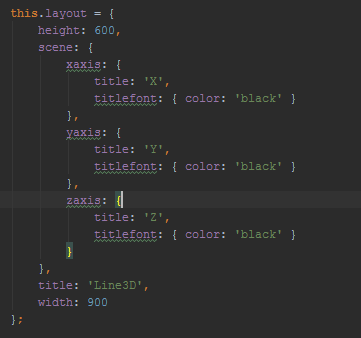


Рис 4.27- Код **layout**

В цьому об’екті ми описуємо ширину та висоту нашої сітки даємо назву нашим векторам та задаємо їм колір.

Після цього ми повинні створити новий плот для відображення площин

(рис 4.28).



Рис 4.28-Створення плота відображення

При створюванні плота ми спочатку повинні показати до якого самого елемента буде прив’язана сітка , масив наших площин які ми будемо відображати та наші настройки сітки. Отже на виході ми отримаємо красиву площину (рис 4.29)

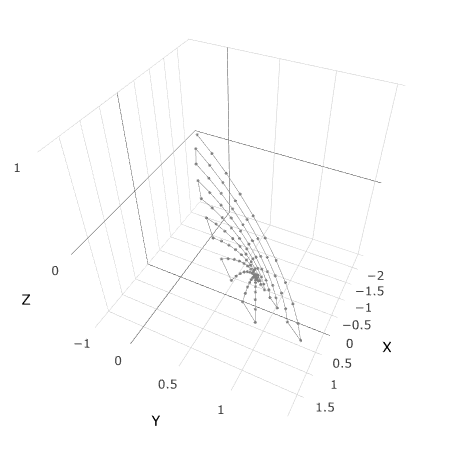


Рис 4.29-Площина **plot1**

Отже мії web-оріентованний додаток буде виглядати ось таким чином (рис 4.30)

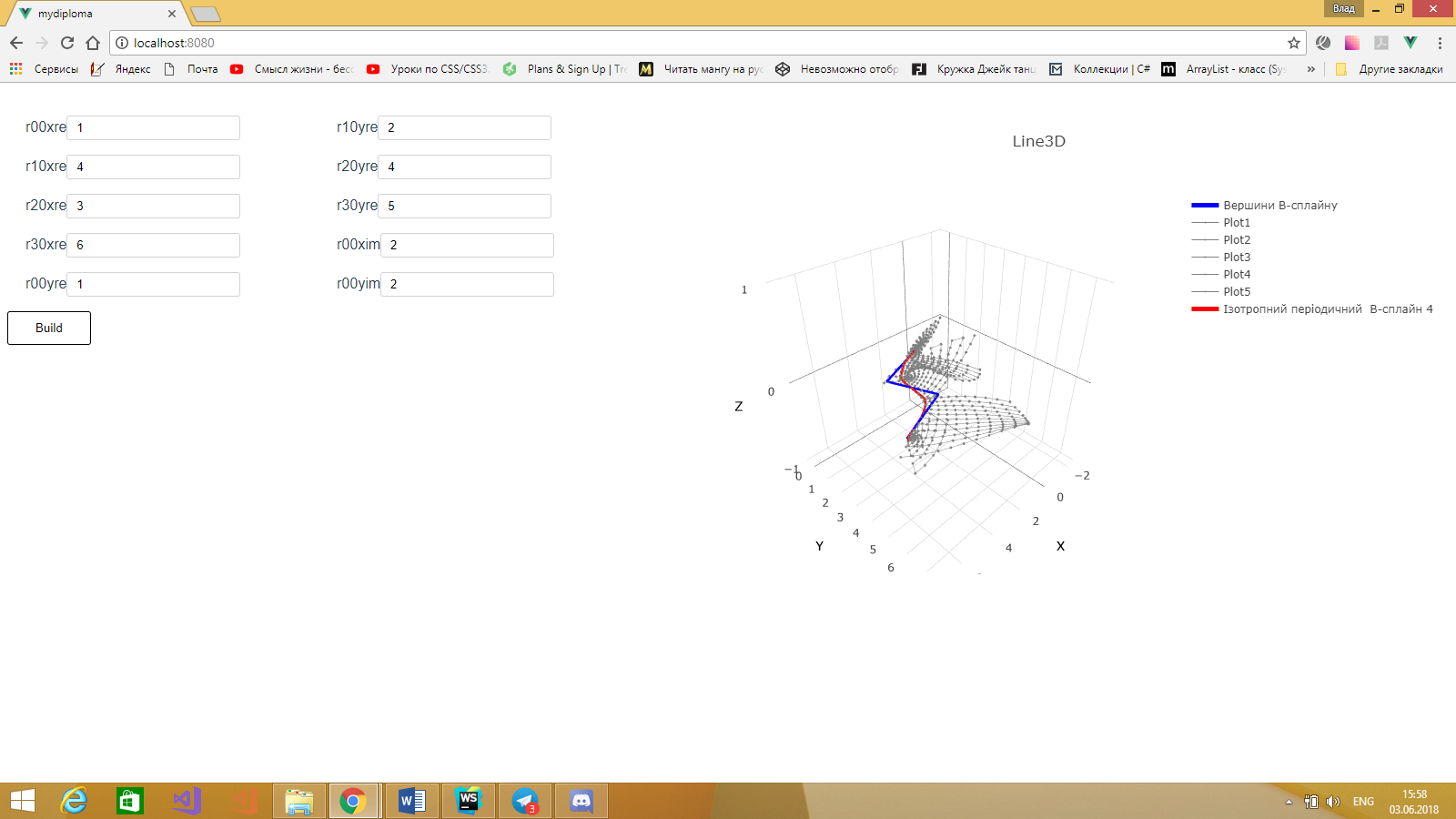


Рис 4.30 - web-оріентованний додаток

Розберемо цей додаток більш детальніше. Ви бачите десять полів для вводу данних (рис 4.31)

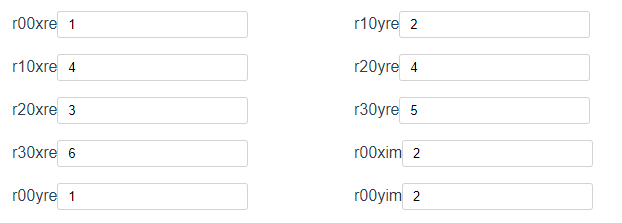


Рис 4.31-Поля для вводу данних

Звісно якщо вхідні данні будуть змінюватися , буде змінюватися і площини (рис 4.32). Для цього необхідно ввести нові данні в поля та натиснути на кнопку Build.

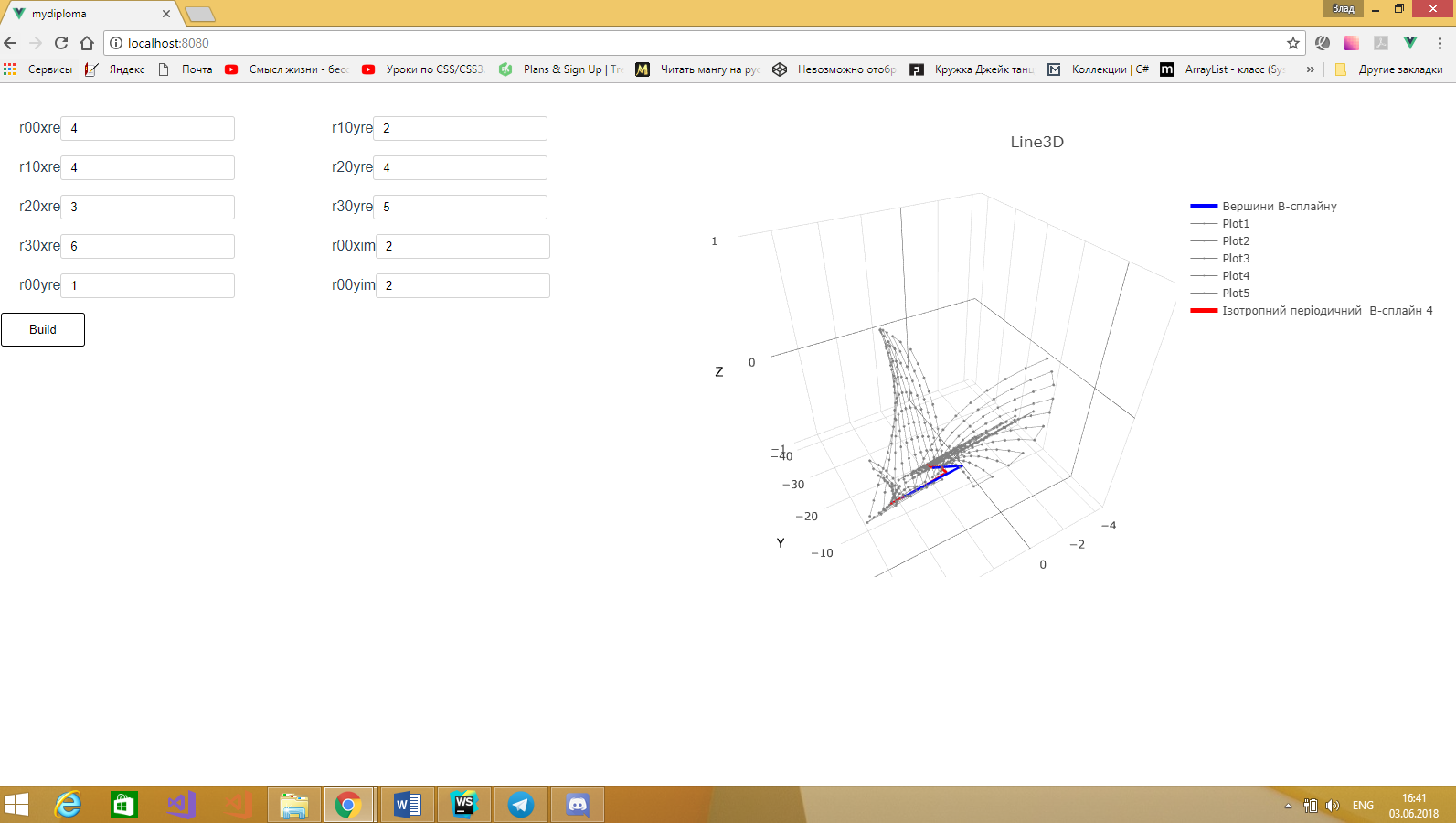


Рис 4.32-Зміненні площини

Також є реалізація відображення одной чи декількох площин. Це легенда яка знаходиться з правої сторони координатної сітки (рис 4.33).

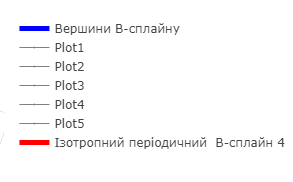


Рис 4.33-Легенда площин

Завдяки цьому можно вибирати для просмотру площини які людина хоче дивитися чи ні , та подивитися як одні площини впливають на інші. Отже якщо людина захоче відключити одну с площин , чи вид В-сплайну, він просто повинен натиснути на це на легенді (рис 4.34).

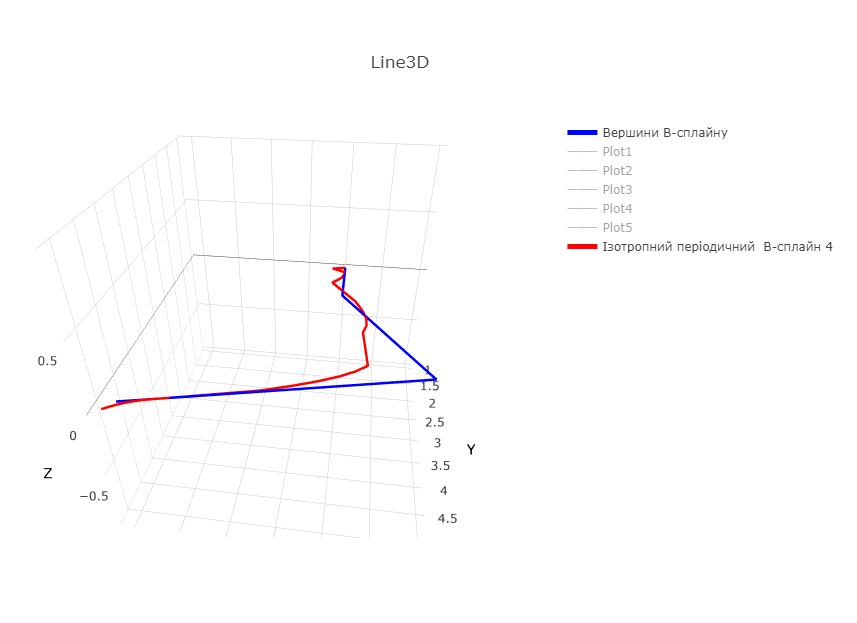


Рис 4.34-Відключення всіх площин окрім В-сплайну

Як ви можити побачити всі площини які не активні на данній момент становитимуся сірого кольору та трішки прозрачні.

Також реалізован спеціальний ункціонал для виведення координат точки при наведенні. Це реалізованно для того щоб користівач міг подивитися більше інформації щодо своеї площини і зрозуміти які саме розміри необхідні для побудування деяких площин (рис 4.35).

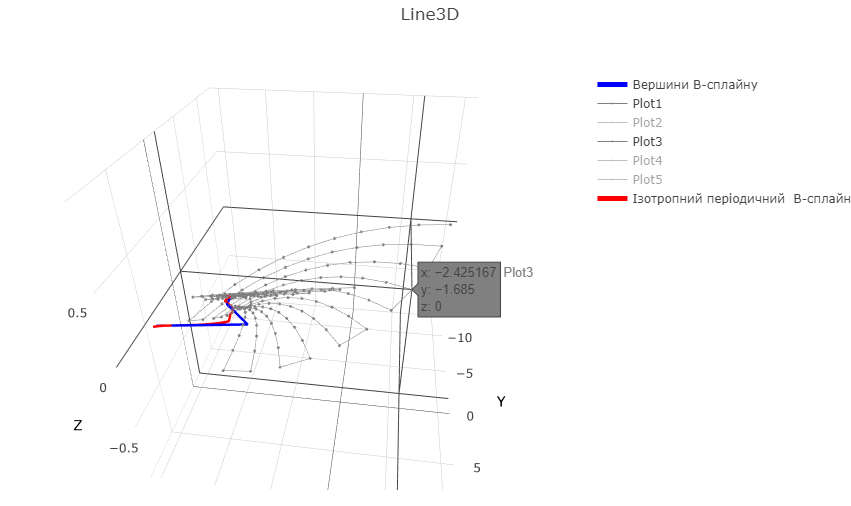


Рис 4.35-Відображення координат точок площин

**ВИСНОВКИ**

Під час проходження практики було покращено навички розробки прогрмних продуктів за допомогою середовища розробки програмного забезпечення VisualStudio Code, мови розмітки гіпертексту HTML, мови опису сторінок CSS та мов програмування JavaScript та його фреймворками Math.js та Plotly.js.

Також були отримані навички застосування принципів реактивного програмування та архітектури клієнт. Було виконано глибоке дослідження клієнт архітектури, різноманітних патернів. Ці технології забезпечили для користувачів можливості створення, редагування.

Використання веб-орієнтованість забезпечили дуже високу доступність програмного засобу та можливість запровадити єдиний інтерфейс для взаємодії з продуктом, незалежно від операційної системи, встановленої на користувацькому пристрої та його програмного оточення.

Таким чином, проходження практики поглибило знання різноманітних технологій розробки клієнт застосунків та взаємодії зі сторонніми сервісами, використання яких є невід’ємною частиною сучасних програмних продуктів.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Бадаєв Ю.І. Застосування кривих четвертого порядку в полікоординатному представленні для опису складних об’єктів Ю.І. Бадаєв, Н.М. Аушева, О.П. Задорожная
2. Аушева Н. М. Моделювання сіток з напрямними дробово-раціональними кривими [Текст] / Н.М. Аушева // Сучасні проблеми архітектури та містобудування: Наук.-техн. збірник**. –** Вип.34. – К.: КНУБА, 2013. – С. 393–399.
3. Аушева Н.М. Моделювання плоских сіток на основі дробово-раціональних ізотропних кривих [Текст] / Н.М. Аушева //Журнал **«Технологічний аудит та резерви виробництва»: Наукові підсумки 2013. – Т.6, №4(14).–Редакція «Східно-Європейського журналу передових технологій». – С. 41–43.**
4. Аушева Н.М. Конструювання криволінійних обводів сегментами кривих п’ятого порядку
5. Vue.js for begginers - <https://habr.com/post/329452/>
6. Vue.js documentation- https://ru.vuejs.org/v2/guide/index.html