УДК 004.02 DOI: 10.31673/2412-4338.2021.041219

Печериця В.В, Бондарчук А.П., Замрій І.В. *Державний університет телекомунікацій, Київ*

РОЗРОБКА МЕТОДИКИ ПРОТОТИПУВАННЯ ОБ'ЄКТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ НА БАЗІ ТЕХНОЛОГІЇ JAVASCRIPT, NODE.JS

Анотація: *Робота присвячена питанням підвищення якості процесу проектування та розробки об'єктів інформаційних систем за рахунок використання прототипування з елементами автогенерації програмного коду на базі технології JavaScript, Node.js. В процесі проектування та створення об'єктів інформаційних систем можуть виникати розбіжності або потреби в змінах, оскільки вимоги можуть динамічно змінюватись в процесі ітерацій проєкту, що призводить до збільшеня затрат часу на розробку. Вирішення подібних проблем передбачає попередню реалізацію певної функціональності майбутньої інформаційної системи чи окремих її частин у вигляді прототипу. Він дає змогу узгодити всі потенційні питання та суперечки ще до етапу розробки програмного коду та допомагає орієнтуватися як команді, так і користувачам на кожному етапі створення програмного забезпечення. Проведено аналіз сучасних підходів до прототипування та інструментальних засобів для їх реалізації. Визначено основні види прототипів, що використовуються на проєктах та їх призначення. Аналіз популярних в корпоративній розробці інструментальних засобів для створення прототипів показав, що ключовою проблемою є відсутність зв'язку прототипу інтерфейсу користувача з кінцевим програмним кодом, що викликає чисельні правки на проєкті, пов'язані зі змінами та не повною узгодженістю всіх вимог до функціональності майбутнього програмного продукту. Запропоновано методику прототипування об'єктів інформаційних систем. Модель методики ґрунтується на архітектурі MVC, але доповнена елементами більш низького рівня, як от логічні оператори, змінні, методи, модуль, що дозволяє здійснювати автогенерацію програмного коду. З використанням вказаної моделі розроблено метод побудови програмного коду на основі прототипу та реалізовано його на базі технології JavaScript, Node.js. Проведене експериментальне дослідження на двох проєктах з розробки інформаційних систем, схожих за характеристиками, продемонструвало скорочення загального часу розробки та зменшення часу на внесення правок при застосуванні розробленої методики.*

Ключові слова: *прототипування, програмне забезпечення, об'єкт інформаційної системи, прототип.*

Pecheritsa V.V., Bondarchuk A.P., Zamrii I.V. *State University of Telecommunications, Kyiv*

DEVELOPMENT OF METHODOLOGY FOR PROTOTYPING INFORMATION SYSTEM OBJECTS BASED ON TECHNOLOGY JAVASCRIPT, NODE.JS

Abstract: *The work is devoted to issues of improving the quality of the design and development of information system objects through the use of prototyping with elements of autogeneration of software code based on JavaScript, Node.js technology. In the process of designing and creating objects of information systems, disagreements or the need for changes may arise, since the requirements may change dynamically during project iterations, which leads to an increase in development time. Solving such problems involves preliminary implementation of certain functionality of the future information system or its individual parts in the form of a prototype. It allows you to agree on all potential issues and disputes before the software code development stage and helps guide both the team and users through each stage of software development. An analysis of modern approaches to prototyping and tools for their implementation was carried out. The main types of prototypes used in projects and their purpose are determined. The analysis of popular tools for creating prototypes in corporate development showed that the key problem is the lack of connection between the user interface prototype and the final software code, which causes numerous revisions to the project, associated with changes and incomplete consistency of all requirements for the functionality of the future software product The method of prototyping objects of information systems is proposed. The methodology model is based on the MVC architecture, but is supplemented with lower-level elements, such as logical operators, variables, methods, a module that allows auto-generation of software code. Using the specified model, a method of building software code based on a prototype was developed and implemented on the basis*

of JavaScript, Node.js technology. The conducted experimental research on two projects for the development of information systems, similar in characteristics, demonstrated a reduction in the total development time and a reduction in the time for making changes when applying the developed methodology.

Keywords: *prototyping, software, information system object, prototype.*

Печерица В.В., Бондарчук А.П., Замрий И.В. *Государственный университет телекоммуникаций, г. Киев*

РАЗРАБОТКА МЕТОДИКИ ПРОТОТИПИРОВАНИЯ ОБЪЕКТОВ ИНФОРМАЦИОННОЙ СИСТЕМЫ НА БАЗЕ ТЕХНОЛОГИИ JAVASCRIPT, NODE.JS

Аннотация *Работа посвящена вопросам повышения качества процесса проектирования и разработки объектов информационных систем за счет использования прототипирования с элементами автогенерации программного кода на базе технологии JavaScript, Node.js. В процессе проектирования и создания объектов информационных систем могут возникать разногласия или потребности в изменениях, поскольку требования могут динамически изменяться в процессе итераций проекта, что приводит к увеличению затрат времени на разработку. Решение подобных проблем предполагает предварительную реализацию определенной функциональности будущей информационной системы или отдельных ее частей посредством прототипа. Он позволяет согласовать все потенциальные вопросы и споры до этапа разработки программного кода и помогает ориентироваться как команде, так и пользователям на каждом этапе создания программного обеспечения. Проведен анализ современных подходов к прототипированию и инструментальных средств для их реализации. Определены основные виды используемых на проектах прототипов и их назначение. Анализ популярных в корпоративной разработке инструментальных средств для создания прототипов показал, что ключевой проблемой является отсутствие связи прототипа пользовательского интерфейса с конечным программным кодом, что вызывает многочисленные правки на проекте, связанные с изменениями и не полной согласованностью всех требований к функциональности будущего программного продукта. Предложена методика прототипирования объектов информационных систем. Модель методики основывается на архитектуре MVC, но дополнена элементами более низкого уровня, такими как логические операторы, переменные, методы, модуль, что позволяет осуществлять автогенерацию программного кода. С использованием указанной модели разработан метод построения программного кода на основе прототипа и реализован на базе технологии JavaScript, Node.js. Проведенное экспериментальное исследование на двух проектах по разработке информационных систем, схожих по характеристикам, продемонстрировало сокращение общего времени разработки и уменьшение времени внесения правок при применении разработанной методики.*

Ключевые слова: *прототип, программное обеспечение, объект информационной системы, прототип.*

1. Вступ

Значна частина в сучасних успішних систем будь-якого призначення (додаток, інструмент програмування, гра) створені на основі прототипу. З підвищенням потреб в нових програмних систем зростає потреба в збільшені швидкості та не зменшенні якості розробки тому потрібно удосконалити вже готові рішення чи створити власний інструмент для прототипування. Проте для початку необхідно зробити збір інформації по всіх популярних існуючих додатків, зробити їх опис та порівняти між собою, що б зробити висновки на основі яких можливо сформувати план та структуру яка дасть змогу поліпшити показники аналогів та створити нову функціональність та підхід.

Після імплементації методики на основі зібраних даних з'являється можливість застосувати її на практиці та сформувати власні метрики швидкості, часу розробки, порівняти з його аналогами та оцінити наскільки продуктивним стало дане рішення, та на основі таких даних додати певні зміни. Такі кінцеві рішення і формують нову методику.

Вищесказане обумовлює актуальність і необхідність проведення досліджень в цьому напрямку.

2. Аналіз літературних джерел і постановка проблеми

В роботі [1] представлена розгорнута класифікація прототипування таких способів, як:

–еволюційне прототипування;

–експериментальне прототипування.

Проте прототипи розділяють ще за категоріями, такі як «Горизонтальні» та «Вертикальні», які мають свої параметри та характеристики [2]. Особлива увага приділяється швидкому прототипуванню [3]. Технології, які використовують в цьому процесі, наприклад:

1. Розробка із застосуванням динамічних мов високого рівня.
2. Використання мов програмування баз даних.
3. Складання додатків з повторним використанням компонентів.

Швидке прототипування реалістичної моделі інтерфейсу користувача додатку дозволяє легко його змінити і залучає користувачів на ранніх етапах розробки додатку. Щоб створити успішний прототип, слід вибрати інструмент швидкого створення прототипів, який відповідає заданим потребам формування невеликого набору команду з розробки прототипів, отримати багато відгуків користувачів і повторювати, доки користувачі не будуть досягнені необхідні показники якості інтерфейсу [4].

Для автоматизації вище описаних процесів використовується програмне забезпечення найрізноманітніших видів, виду інтерфейсу та призначення в застосовувані. Вони мають свої переваги та недоліки [5, 6].

Ефективне створення прототипів для розробників програмного забезпечення – це інформативний ресурс, який допоможе будь-кому (незалежно від того, має хтось художній талант, доступ до спеціальних інструментів чи вміння програмувати) використовувати гарний стиль, методи та інструменти для створення прототипів та ефективного керування ними. створення прототипів. Модель прототипування – це модель розробки програмного забезпечення, в якій прототип будується, тестується та переробляється до тих пір, поки не буде досягнутий прийнятний прототип. Це також створює базу для виробництва остаточної системи або програмного забезпечення і найкраще працює в сценаріях, де вимоги проекту невідомі в деталях. Це своєрідний ітеративний метод проб і помилок, який відбувається між розробником і клієнтом [7].

Для імплементації створеної методики прототипування традиційно використовують вже готові технології, які забезпечують необхідні показники якості. Для запуску серверу, який потрібен для роботи JavaScript коду, використовується платформа Node.js, [8]. Framework дозволяє прийняти, обробити, надіслати дані клієнтській частині додатку [9]. Клієнтська частина створюється окремо, вона реалізує в собі користувацький інтерфейс що задовольняє ключовим характеристикам: надійність, простота, інтерактивність, кількість плагінів [10].

Аналіз підходів до прототипування та існуючих технологій, що їх реалізують, показав, їх недостатню функціональність, яка б дозволила задовільнити ключові потреби процесу прототипування та сформувати прототипи із заданими показниками якості. Затребуваним є підхід до прототипування, оптимізований з точки зору показників швидкості та простоти розробки.

3. Мета і задачі дослідження

Об’єкт дослідження – процес проектування та розробки об’єктів інформаційних систем.

Предмет дослідження – методи та інструментальні засоби для проектування та розробки об’єктів інформаційних систем.

Мета дослідження – підвищення якості процесу проектування та розробки об’єктів інформаційних систем за рахунок використання прототипування з елементами автогенерації програмного коду.

Досягнення визначеної мети реалізується через виконання наступних задач:

– аналіз підходів до прототипування та інструментальних засобів для їх реалізації;

– розробка моделі та методу прототипування об’єктів інформаційних систем;

– імплементація розробленої методики в процес корпоративної розробки на базі технології JavaScript, Node.js.

4.1 Аналіз підходів до прототипування та інструментальних засобів

Прототипування (prototyping) - це один із найважливіших та в основному найвикористовуваніших сучасних інструментів для формування потреб та вимог. Прототипи будуються для відображення системи або чи переважаючої її основної частини для користувачів з метою формування потреб. Прототип формує в собою демонстраційну (базову) систему – «нашвидкуруч і грубо» створену робочу модель рішення, яка демонструє взаємодію з її елементами і моделює поведінку системи при взаємодії з нею при ініціалізації з користувачем чи іншою системою різних подій. Складність та зростаючі вимоги, які формуються зі зростанням розвитку сучасних систем, які створюють все більший попит на прототипування та є майже невід'ємним компонентом у створені інформаційної системи або її частин. Прототипи дають змогу досить оцінити можливість її реалізації та користь системи ще до початку її створення.

У переважній більшості ситуацій, прототип – це метод виявлення вимог, які складно отримати від замовника за допомогою інших інструментів. Часто такі моменти можливо побачити на системах, яким необхідно надати у користування користувачам (чи іншим системам) нову функціональність. Також дана ситуація є характерна у тих випадках, у випадку суперечливих вимог, а також при проблемах комунікації між замовником (користувачем) і розробниками.

В залежності від призначення та перспектив подальшого використання прототипів, розрізняють:

–архітектурний прототип – використовується для демонстрації можливостей системи;

–прототип вимог – це часткова реалізація програмного забезпечення, створена з метою допомогти розробникам, користувачам краще зрозуміти вимоги до системи;

–одноразовий – передбачається, що створюється макет, який використовується тільки на певному етапі, але не стане частиною готової системи;

–еволюційне прототипування - ставить за мету послідовно створювати макети системи, які будуть все ближче і ближче до реального програмного забезпечення;

–вертикальні прототипи - втілюють зріз функціональності застосування (додатку) від інтерфейсу користувача до сервісних функцій;

–горизонтальний прототип – прототип в якому не реалізуються всі шари архітектури і нюанси системи, але втілюються деякі особливості інтерфейсу користувача.

В таблиці 1 наведено порівняльний аналіз інструментальних засобів прототипування, які широко використовуються в сучасних IT-проєктах для візуалізації функцій користувача через інтерфейсні елементи. Слід зазначити, що прототипування на рівні інтерфейсу в цілому відноситься до горизонтального типу, але зазначені технології, тим не менш, дозволяють оцінити наявну функціональність додатку за рахунок інтерактивності створюваних прототипів. Значною перевагою подібних інструментів прототипування є можливість організовувати віддалений доступ всіх стейкхолдерів проєкту до прототипу, що позитивно сприяє процесу узгодження вимог.

Таблиця 1

Порівняльна характеристика інструментальних засобів для прототипування

Переваги	Недоліки
Axure RP Pro	
Стандартна структура системи, яка подібна до Microsoft Office, Microsoft Visio, що зменшує затрати на ознайомлення з програмним забезпеченням. Інтерактивність створеного прототипу(що поліпшує користувацький досвід).	Застосування обмежене тільки створенням візуального інтерфейсу. У системі не передбачено роботу кількох користувачів з одним проектом на сервері. Немає автогенерації коду.

iRise	
Широкий набір функціональних можливостей Можливість інтеграції з різними системами (jama, Jira, Azure DevOps).	Слабка інформаційна підтримка у вільному доступі Велика вартість. Немає автогенерації коду.
Sketch	
Наявність інструментів експорту графіки. Інтеграція з нативними пристроями	Можна установити тільки на macOS. Виникають проблеми з файлами великих розмірів
Figma	
Дозволяє працювати одночасно над одним проектом багатьом користувачам. Працює в браузері та не потребує обов'язкової установки. Система підтримує всі найпопулярніші ОС.	Обмежена кількість безкоштовних проектів. Потребує постійного підключення до інтернету.

4.2 Розробка моделі та методу прототипування

Оскільки прототип – це не повноцінна система, а тільки його схематична модель, яка уособлює тільки його каркас чи макет, то команда розробки орієнтується на нього на початку розробки, щоб мати певний погляд на готову систему. Оскільки більшість систем прототипування сформовані за принципом переносу та накладання графічних блоків та компонентів, які візуально формують певний макет прототипу, даний підхід було взято за основу в розроблюваній системі. В якості засобу розробки використано Vue.js, який має досить низький поріг входження для новачків, його крива навчання дозволяє майже будь-якому програмному інженеру в невеликі строки вивчити документацію та починати писати проекти, такі переваги доповнює величезна кількість готових пакетів(модулів), що допомагає суттєво скоротити час розробки front-end частини системи.

Ключовою проблемою проаналізованих систем є відсутність автогенарації коду по вже сформованих графічних блоках які і формують логіку прототипу, це суттєво збільшує час на програмну реалізацію форм інтерфейсу. Автогенерація потребує взаємодії з файловою системою проєкту. Для цього було застосовано фреймфорк Nest.js який створений для розробки серверних об'єктів інформаційних систем. Для запуску вище описаних інструментів використовується платформаNode.js – це кросплатформне середовище виконання JavaScript із відкритим вихідним кодом, яке працює на рушії V8 і виконує код JavaScript за межами веб-браузера. Node.js дозволяє використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера – запуск сценаріїв відбувається на стороні сервера для створення динамічного вмісту веб-сторінки, перш ніж сторінка буде відправлена у веб-браузер користувача. Такий варіант дизайну системи спрямований на оптимізацію пропускної здатності та масштабованості у веб-додатках з багатьма операціями введення/виводу, а також для веб-додатків реального часу (наприклад, комунікаційні програми в режимі реального часу та браузерні ігри).

Розглянемо структуру системи прототипування. Вся основна функціональність міститься на сервері. Він містить в собі логіку взаємодії з клієнтом, зберігає всі файлові структури з файлами конфігурації прототипів, які містять параметри для генерації вихідного коду, який в подальшому можливо і потрібно використовувати, як фасад для повноцінного проекту. Сама структура прототипу уособлює в собі реалізацію MVC (модель, контроллер, сервіси). Для генерації коду було прийнято рішення виділити в моделі прототипування сутності більш низького рівня:

– $< LogOper >$ логічні оператори - дають змогу за заданими параметрами виконати дії чи зупинити програму;

– $< Variables >$ змінні - зберігають інформацію чи посилання на неї, за основу береться синтаксис мови JavaScript;

– $< Methods >$ методи – відповідають поняттю методу в об’єктно-оріжнтованому програмуванні, можуть мати властивості синхронності чи асинхронності тощо, та відповідають за логіку додатку;

– $< Module >$ модуль –структура, яка слугує обгорткою для об'єктів системи, вказаних вище, зберігає та структурує їх в окремому файлі та інкапсулює його тіло.

Таким чином, модель прототипу можна описати виразом (1):

$$P = \{< LogOper >, < Variables >, < Methods >, < Module >\}. \quad (1)$$

Для системи прототипування принциповим питанням є взаємодія з базою даних, оскільки доступ до неї необхідний як в процесі прототипуваня так і при її запуску, до того ж всі дані її маю бути ідентичні до об’єктів інформаційної системи, представлених і прототипі інтерфейсу, на всіх етапах. Дану задачу виконує окремий клас. Він отримує в свій конструктор шлях до файлу з бази даних. Після цього він за допомогою драйверу підключається до документу з розширенням .sqlite, яке містить всі дані по проекту. Файл конфігурації прототипу має наступну структуру:

–/config - змінні середовища та конфігурація;
–/controllers - містить параметри контролеру;
–/db - параметри бази даних;
–/models - моделі бази даних;
–/routes - маршруту для кінцевих;
–/services - допоміжні методи побудови логіки;
–/utils - утилітарні класи;
–db.sqlite - зебрежена інформація в базі даних;
–app.json - точка входу в додаток.

Кожна директорія та її назва відповідає назві сутності прототипу, що полегшує та покращує навігацію по проєкту.

На рис.1 наведено етапи методу автогенерації коду на основі прототипу. Їх реалізовано у вигляді скрипту, який використовується для створення JavaScript-коду на основі о'бєктів, що зберігають в собі визначення сутностей, його типів та підтипів, за якими в свою чергу визначаються методи.

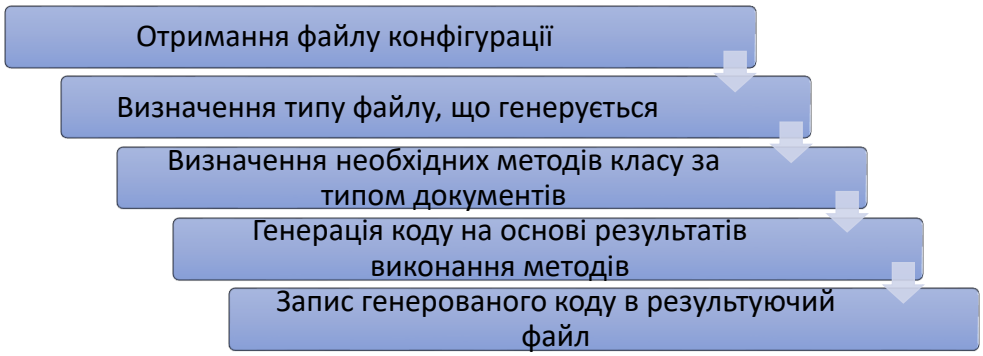

Рис. 1. Етапи методу автогенерації коду на основі прототипу

4.2 Застосування методики в корпоративній розробці

Для перевірки властивостей систем, розроблених на основі запропонованої методики, було створено два проєкти «Система управління працівниками» та «Система управління та моніторингу інвентаря» з наступними характеристиками: сервер створений за архітектурним стилем REST, для створення системи використовувалась мова JavaScript, клієнтська частина реалізована з використанням Vue.js. Члени команди розробки систем є однаковими на обох проєктах: Manager; Back-end розробник; Front-end розробник; дизайнер; тестувальник. Відмінність проєктів полягала в їх призначенні та внутрішній логіці. Ключова

функціональність проєкту «Система управління працівниками»: моніторинг та створення задач для користувачів, формування звітів. Ключова функціональність проєкту «Система управління та моніторингу інвентаря»: відслідковування наявних інструментів; закупка інструментів та інвентарю. Кількість завдань при розробці кожної системи відрізняється не більше ніж на 3%, і, зважаючи на те, що обсяги окремих завдань планувались приблизно однаковими в обох системах, це дозволяє зробити висновок стосовно можливості порівняння загального часу розробки систем з використанням традиційного підходу та запропонованої методики.

В таблиці 2 наведено витрати часу на створення проекту у порівнянні зі звичайним підходом до прототипування та запропонованою методикою. Зірочкою помічено етапи, в яких використовувалась методика.

Таблиця 2

Результати застосування методики в корпоративній розробці

Назва системи	Система управління працівниками	Система управління та моніторингу інвентаря
Закладений час на розробку	320 годин	325 годин
Кількість завдань	52	54
Обговорення	20 годин	15 годин
Прототипування	20 годин	13 годин
Дизайн	40 годин + 4 (правки)	38 годин + 3 (правки)
Front-end розробка	140 годин + 24 (правки)	120 годин + 12 (правки)
Back-end розробка	120 годин + 32(правки)	135 годин + 16(правки)
Всього	**400 годин розробки**	**352 годин розробки**

5. Висновки.

Розроблено методику для створення прототипів, яка на відміну від існуючих популярних підходів включає підсистему автоматизованої генерації коду, що дозволяє забезпечити ряд переваг при створенні IT-проєктів, зокрема, забезпечити скорочення загального часу виконання проєкту, а також кількість правок, що вносяться командою в результаті зміни прототипу. Експериментальне дослідження на схожих за ключовими характеристиками проєктах показало зменшення загальних витрат часу на створення системи на 12%. При цьому, перевищення планового часу виконання при використанні традиційних засобів становить 25%, а при застосуванні розробленої методики 8%, що для тривалих проєктів може становити значні величини витрат робочого часу команди. Кількість часу, що було витрачено на правки на етапах, пов’язаних безпосередньо з розробкою програмного коду, вдвічі менше на проєкті з автогенерацією коду, що дозволяє зробити висновки про більш високі показники якості інформаційної системи: більша відповідність реалізованих функцій вимогам користувачів, менша кількість помилок при переході від концепту інтерфейсу до його реалізації.

Подальші дослідження лежать в площині використання методики для створення систем з більш різноманітними характеристиками. Основне обмеження використання методики на поточний момент – це можливість формування тільки серверної частини системи, яка реалізовує REST принципи, однак, це зауваження вирішується імплементацією інших архітектур. Слід зазначити, що дана методика прототипування може аналогічним чином імплементувати клієнтську частину системи.

Список використаної літератури

1. Arnowitz, Jonathan, Michael Arent, and Nevin Berger. *Effective prototyping for software makers*. Elsevier, 2010.

2. Дьоміна А. О. Особливості прототипування об’єктно-орієнтованих програмних систем// Наукові записки НаУКМА. 2012. Т. 138 : Комп'ютерні науки. С. 68-75.

3. N. M. Devadiga, "Tailoring architecture centric design method with rapid prototyping," *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, 2017, pp. 924-930, doi: 10.1109/CESYS.2017.8321218.

4. Tizkar, Ali & M. Tabatabaei, Naser. (2009). Rapid Prototyping for Software Projects with User Interface. Scientific Bulletin of University of PITESTI, Electronics and Computer Science Series. 2. 85.

5. Рудніченко, Микола, et al. "Розробка проекту кроссплатформеної розподіленої інформаційної системи прототипування зовнішнього вигляду програмних застосувань." *Інформаційні технології та суспільство* " (1) (2021): 43-50.

6. Характеристика сучасних систем прототипування. Режим доступу: https://posibniki.com.ua/post-avtomatizaciya-prototipuvannya--informaciynih-sistem

7. Arnowitz, Jonathan & Arent, Michael & Berger, Nevin. (2006). Effective Prototyping for Software Makers (Interactive Technologies).

8. Node.js v18.3.0 documentation. Режим доступу: https://nodejs.org/dist/latest-v18.x/docs/api.

9. Nest.js. Режим доступу: https://docs.nestjs.com

10. Vue.js. Режим доступу: https://vuejs.org/guide/introduction.html

References

1. Arnowitz, Jonathan, Michael Arent, and Nevin Berger. *Effective prototyping for software makers*. Elsevier, 2010.

2. Dyomina, A. O. Peculiarities of prototyping object-oriented software systems// Scientific notes of the Ukrainian Academy of Sciences. 2012. Vol. 138 : Computer Science. C. 68-75.

3. N. M. Devadiga, "Tailoring architecture centric design method with rapid prototyping," *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, 2017, pp. 924-930, doi: 10.1109/CESYS.2017.8321218.

4. Tizkar, Ali & M. Tabatabaei, Naser. (2009). Rapid Prototyping for Software Projects with User Interface. Scientific Bulletin of University of PITESTI, Electronics and Computer Science Series. 2. 85.

5. Rudnichenko, Mykola, et al. "Development of a project of a cross-platform distributed information system for prototyping the appearance of software applications." *Information technologies and society* " (1) (2021): 43-50.

6. Characteristics of modern prototyping systems. Access mode: https://posibniki.com.ua/post-avtomatizaciya-prototipuvannya--informaciynih-sistem

7. Arnowitz, Jonathan & Arent, Michael & Berger, Nevin. (2006). Effective Prototyping for Software Makers (Interactive Technologies).

8. Node.js v18.3.0 documentation. Access mode: https://nodejs.org/dist/latest-v18.x/docs/api.

9. Nest.js. Access mode: https://docs.nestjs.com

10. Vue.js. Access mode: https://vuejs.org/guide/introduction.html