Выполнил: Адамович Константин, 286 группа.

**Отчет по практической работе № 4**

**«Выбор средств реализации»**

**Цель работы**: выбрать средства реализации учебной практики

**Формируемые компетенции:** ПК 5.4 «Производить разработку модулей информационной системы в соответствии с техническим заданием», ПК 5.1 «Собирать исходные данные для разработки проектной документации на информационную систему»; ПК 5.2 «Разрабатывать проектную документацию на разработку информационной системы в соответствии с требованиями заказчика»

**Решение задач:**

1. **Произвести обзор инструментов для создания макетов сайтов.**

Таблица 1 – Сравнение средств прототипирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Скорость прототипирования страницы | Генерация html кода и стилей | Необходимость знания HTML, CSS | Поддержка библиотек | Визуализация |
| Прототип-ирование в коде  (Flutter) | Средняя | да | нет | Flutter пакеты | полностью  готовый код |
| Figma | Средняя | частично | нет | нет | Макет и стили |
| Photoshop | Низкая | нет | нет | нет | Только макет |

Flutter не требует инструмента макетирования так-как имеет обширный набор встроенных элементов и позволяет макетировать “на лету” при написании кода.

При этом скорость прототипирования остается высокой из-за очевидных названий различных объектов интерфейса и подсказок в IDE.

При компиляции используется движок dart2js который генерирует html/css/js код следственно отпадает необходимость знания html/css/js.

Dart огромным списком пакетов (библиотек) для поиска подходящих можно используется поиск для совместимых с web реализацией пакетов.

1. **Выбрать средства реализации фронтенда.**

Фреймворк Flutter (на языке Dart), набор элементов Material (для стиля Material Design)

Плюсы:

1. Компиляция в JS
2. Обширный набор готовых элементов
3. Высокая скорость проектирования интерфейса
4. Язык Dart
5. Удобный пакетный менеджер
6. Возможность компиляции для Android, IOS, Windows, MacOS, Linux при необходимости
7. Адаптивный интерфейс «из коробки»

Минусы:

1. Низкая производительность (по сравнению с нативными решениями)
2. **Выбор базы данных.**

Таблица 1 – Сравнение баз данных

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерии | MySql | MariaDB | Redis | MongoDB | Neo4j | Cassandra | PostgreSQL | SQLite | memchached | ClickHouse |
| Вид базы данных | реляционная | реляционная | резидентная | документоориентированная | графовая | NoSQL | реляционная | реляционная | Хеш-таблица | колоночная |
| Для каких данных используется | Таблица | Таблица | Словари | Документы | графы | массивов данных | Таблица | Таблица | Кэш | Таблица |
| Максимальный  размер базы  данных | Нет ограничения | бесконечность | бесконечность | бесконечность | Размер оперативной памяти | бесконечность | бесконечность | 140 ТБ | Размер оперативной памяти | бесконечность |
| Максимальный размер кластера | 32 эксабайт | - | - | - | - | - | 32 Тбайт | - | - | - |

How do you even create such big json? : ProgrammerHumorИспользовать базу данных не эффективно т.к. основной объем данных файлы которые эффективно хранятся на диске. А конфигурацию сохранять в небольших json файлах.

**4. Выбор языка реализации и фреймворка (если необходимо) для реализации**

**бэкенда.**

Для реализации выбран язык Dart по нескольким критическим причинам:

1. Единая кодовая база с фронтендом
2. Скорость работы (Dart vs Node.js: сравниваем производительность на реализациях HTTP сервера // Хабр URL: <https://habr.com/ru/post/435014/> (дата обращения: 19.05.2020).)
3. Статическая типизация с возможностью использования динамического типа
4. Компиляция под различными системами в нативные исполняемые файлы
5. Быстрое развертывание на сервере
6. Большое количество пакетов
7. Огромное количество концепций, заложенных в язык
8. Высокая скорость разработки
9. Активное развитие языка
10. Открытый исходный код
11. Настройка анализатора синтаксиса языка (Customizing static analysis // dart.dev URL: https://dart.dev/guides/language/analysis-options (дата обращения: 19.06.2020).)
12. Проработанная документация

Планируется использовать фреймворк aqueduct (AQUEDUCT // aqueduct.io URL: https://aqueduct.io/ (дата обращения: 19.06.2020).) который предполагает создание высоканагруженных REST-API с встроенной интеграцией базой данных с готовый аутификацией.

Список используемых источников:

1. Dart vs Node.js: сравниваем производительность на реализациях HTTP сервера // Хабр URL: https://habr.com/ru/post/435014/ (дата обращения: 19.05.2020)
2. Customizing static analysis // dart.dev URL: https://dart.dev/guides/language/analysis-options (дата обращения: 19.06.2020)