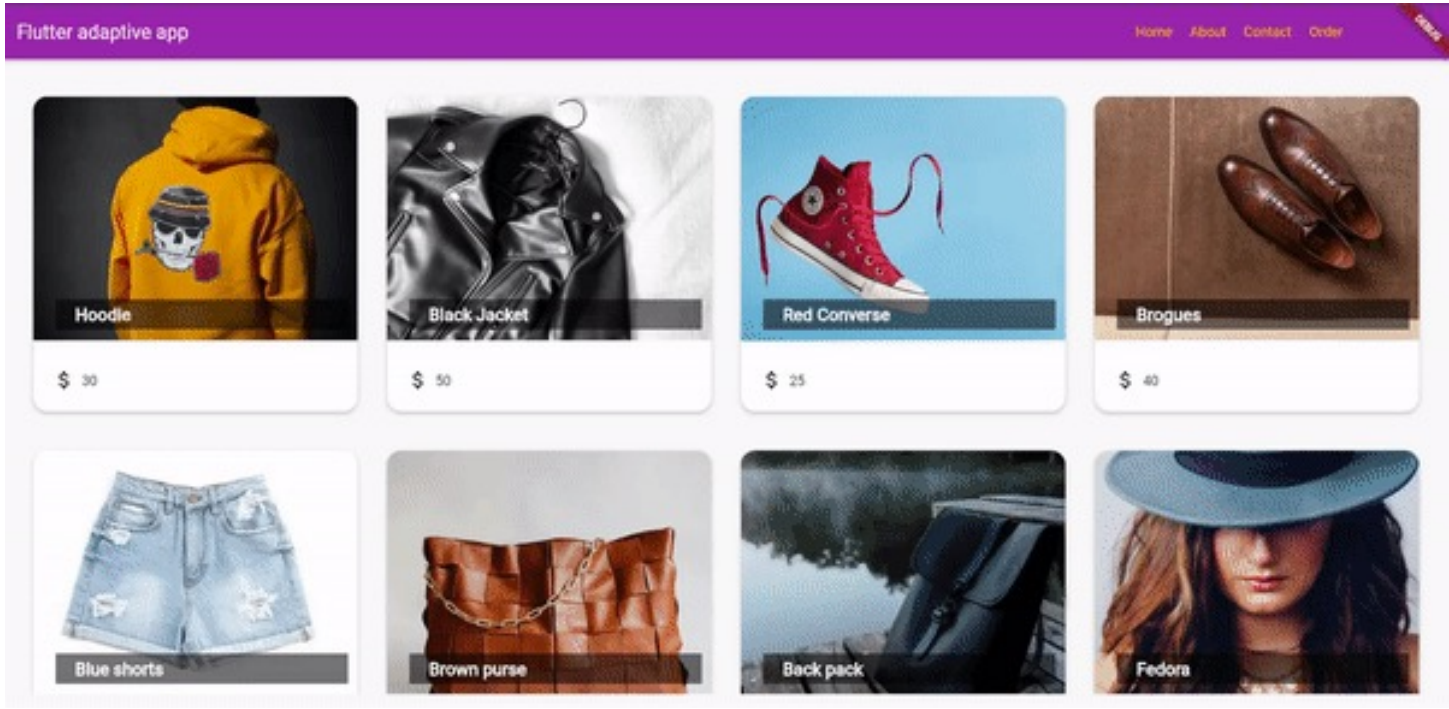
Лекция 11
Развертывание и адаптивность

Разработка интернет приложений

Канев Антон Игоревич

Адаптивность

- свойства обертки, которые позволяют переносить элементы на новую строчку, если предыдущая заполнилась (**flex-wrap**), а так же задают отступы между соседними элементами сверху и снизу (**gap**)
- на свойства самой карточки: в данном случае нас интересует первое свойство **flex**, а точнее последнее значение в нем. Это значение определяет, в какой момент элементы переносятся на новую строчку, а именно если размер элемента становится равным **300px**




```html
<div class="cards__wrapper">
    <div class="card__item">
        <div class="card__img">
            <img src="./images/book.jpg" alt="book">
        </div>
        <div class="card__info">
            <div class="card__text">
                <div class="card__title">Название книги</div>
                <div class="card__description">Описание книги</div>
            </div>
            <button class="card__btn">Приобрести</button>
        </div>
    </div>
</div>
```

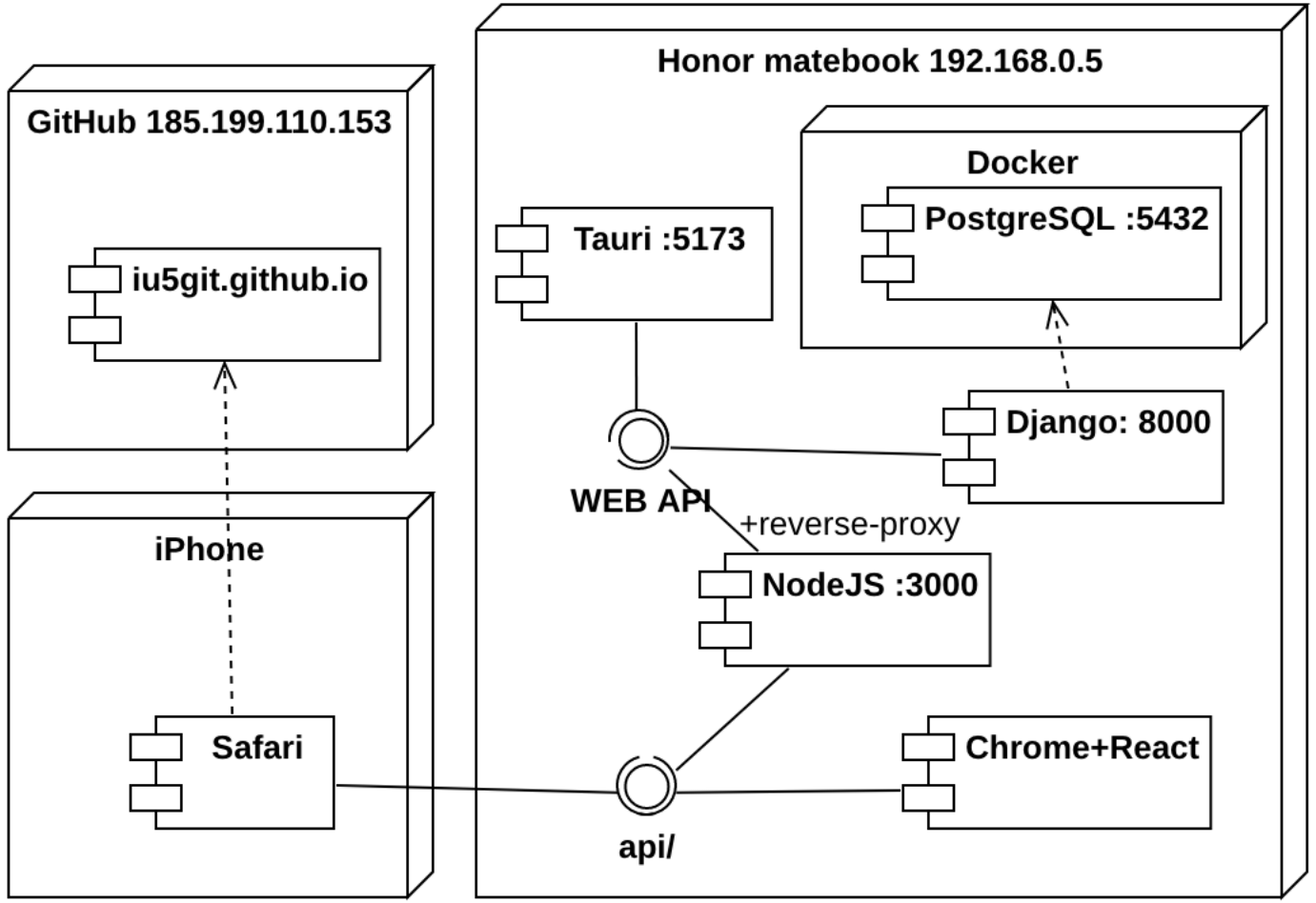
```css
.cards__wrapper {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
    gap: 20px;
}

.card__item {
    flex: 1 1 300px;
    padding: 15px;
    background-color: bisque;
    border-radius: 10px;
    display: flex;
    justify-content: space-between;
}
```

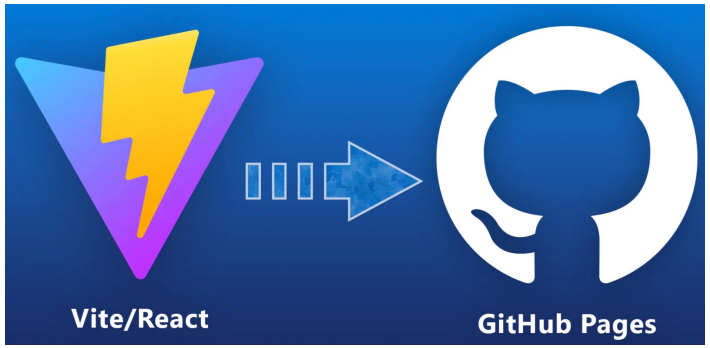
Развертывание

- На **диаграмме развертывания** мы указали наши бэкенд и фронтенд, а также **обратный прокси**, **нативное** приложение и **Pages** – знать отличия!
- Указали IP трех разных устройств и номера портов, которые используются всеми нашими приложениями

- Напрямую к веб-сервису могут обращаться наше нативное приложение и прокси
- Требуется добавить **Minio**, **Redis**
- **Нужно изменить** IP адреса, порты, технологии, названия устройств, а также добавить доп. ДЗ

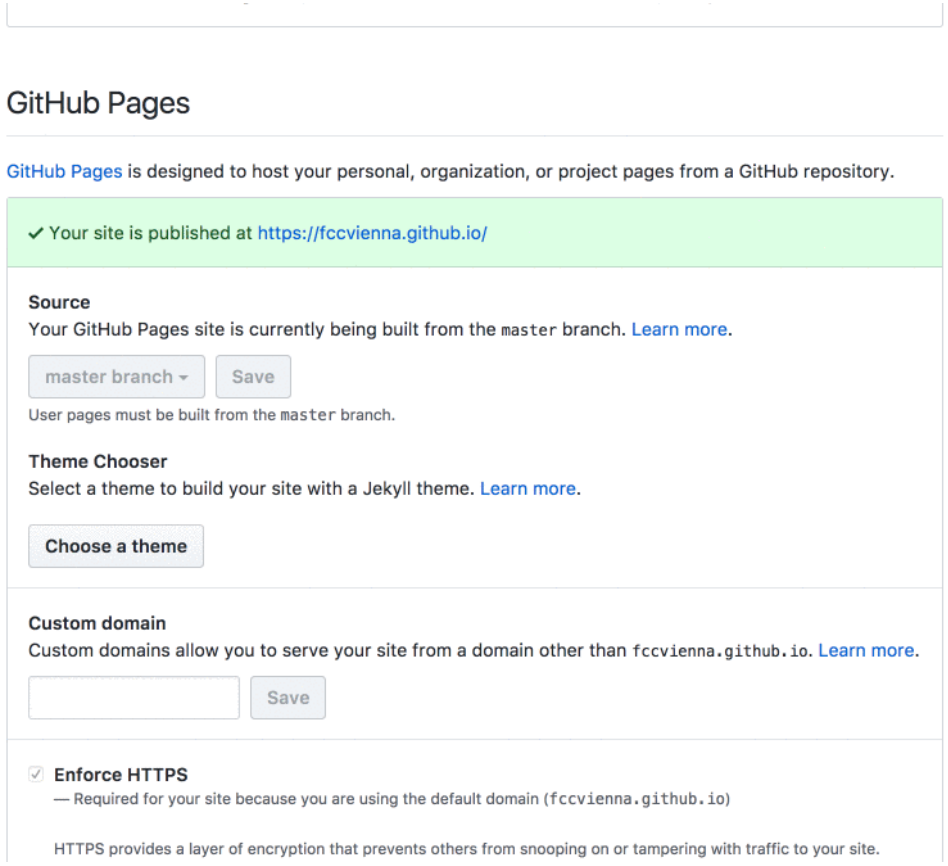


GitHub Pages



https://rashidshamloo.hashnode.dev/deploying-vite-react-app-to-github-pages

- Мы можем бесплатно развернуть наше React приложение на GitHub Pages
- Необходимо выполнить настройки в GitHub: Settings/Pages
- Также необходимо настроить развертывание Vite в проекте



GitHub Pages

- **1. Установить gh-pages пакет**

npm install gh-pages --save-dev

- **2. В файле package.json добавить строки перед "build": "vite build",**

"predeploy": "npm run build",
"deploy": "gh-pages -d dist",

- **3. В файле vite.config.js добавить строку перед plugins: [react()],**

base: "YOUR_REPOSITORY_NAME",

- **4. Выполнить развертывание/обновление**

npm run deploy

Теперь есть ветка gh-pages в репозитории, а приложение развернуто в GitHub

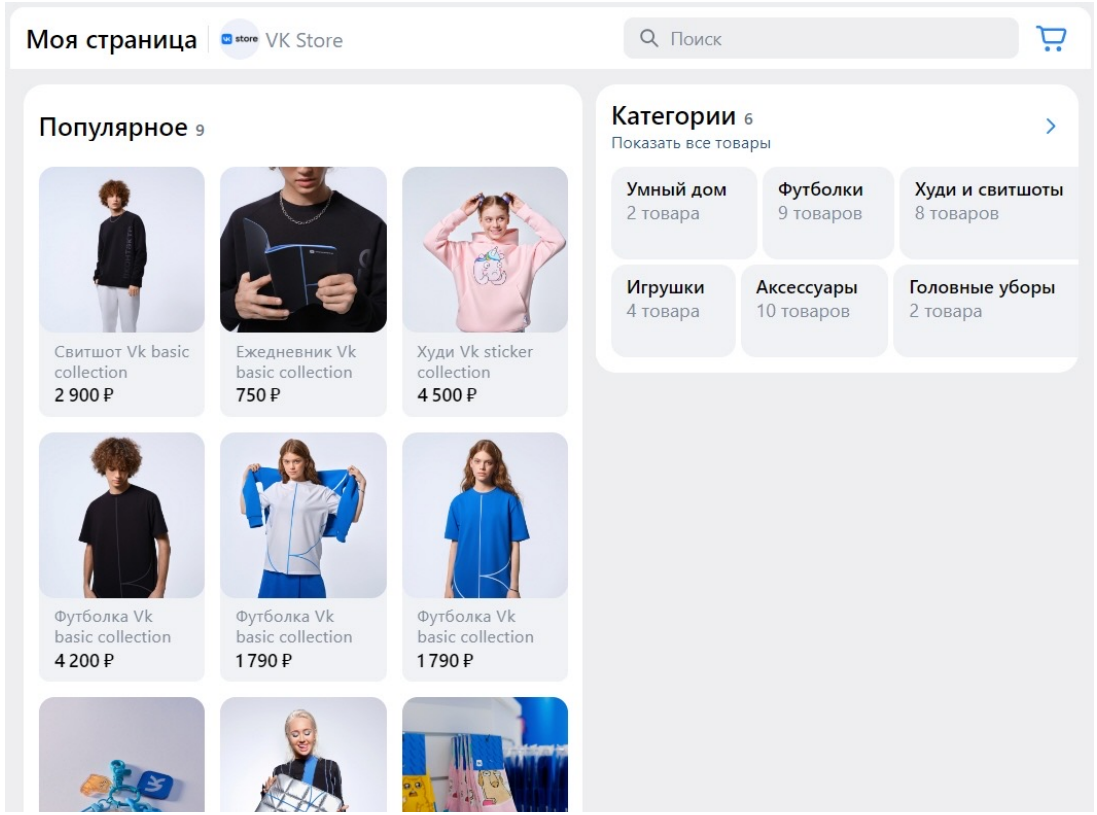
https://rashidshamloo.hashnode.dev/deploying-vite-react-app-to-github-pages

VK mini App

- Создаем React-приложение, но используем бэкенд VK

Магазин

- React-приложение, которое показывает совместную работу различных библиотек ВКонтакте в мини-приложении, а также демонстрирует лучшие подходы к созданию мини-приложений.
- Мини-приложение: vk.com/app51654068
- Исходный код: github.com/VKCOM/vk-mini-apps-examples



Файловая структура

Для удобной работы и упрощения поддержки кода важно правильно организовать исходные файлы. В нашем примере файлы приложения сгруппированы в следующие папки:

Папка	Описание
`api`	Код, который выполняет API-запросы. Его можно легко заменить без необходимости менять код других частей приложения.
`components`	Код React-компонентов.
`pages`	Код страниц, составленных из React-компонентов.

WebView и iframe

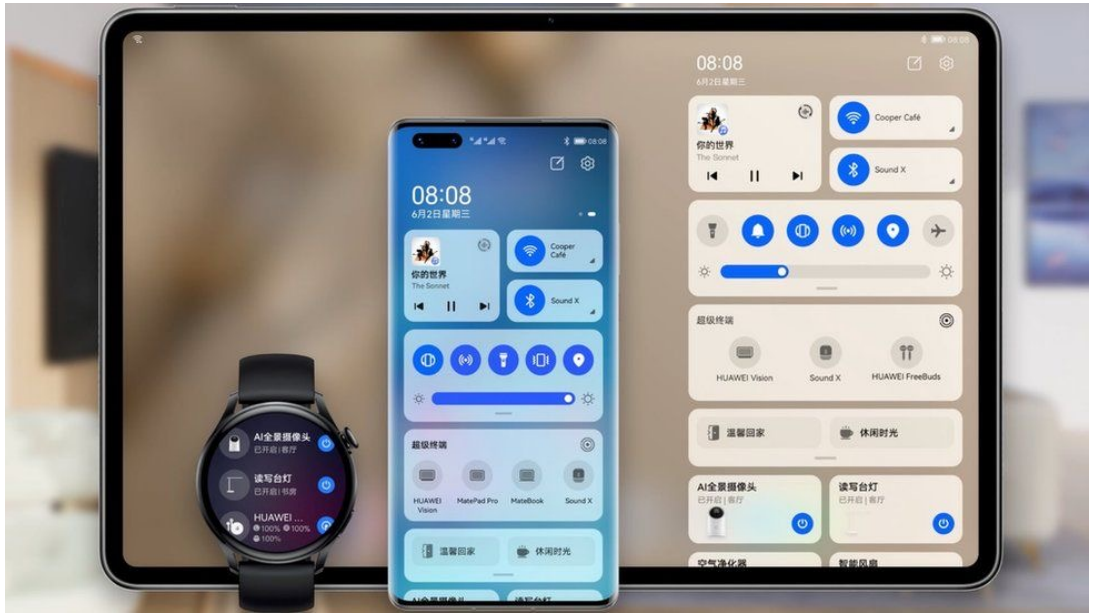
Две важные веб-технологии:

- **WebView** – компонент, позволяющий использовать веб (верстку, стили, код JS) внутри приложений
- **iframe** – для встраивания другого HTML-документа в текущий: карта, видео, пост из соцсети

```
<iframe src="https://www.google.com/" height="500px" width="500px"></iframe>
```

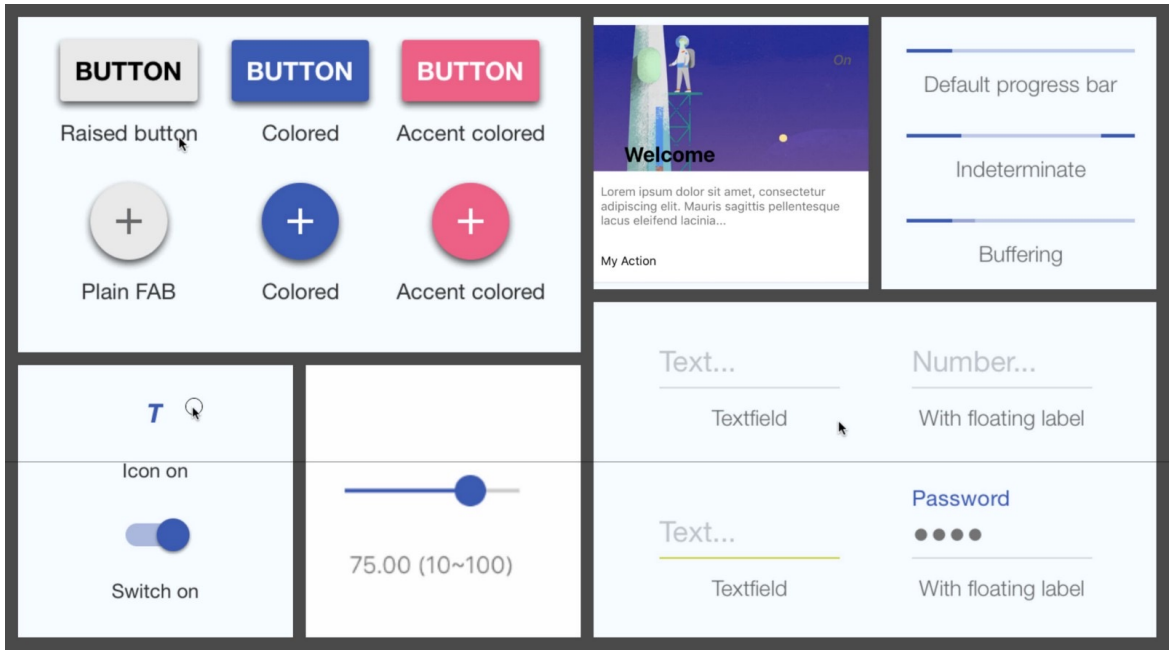
Виды нативных приложений

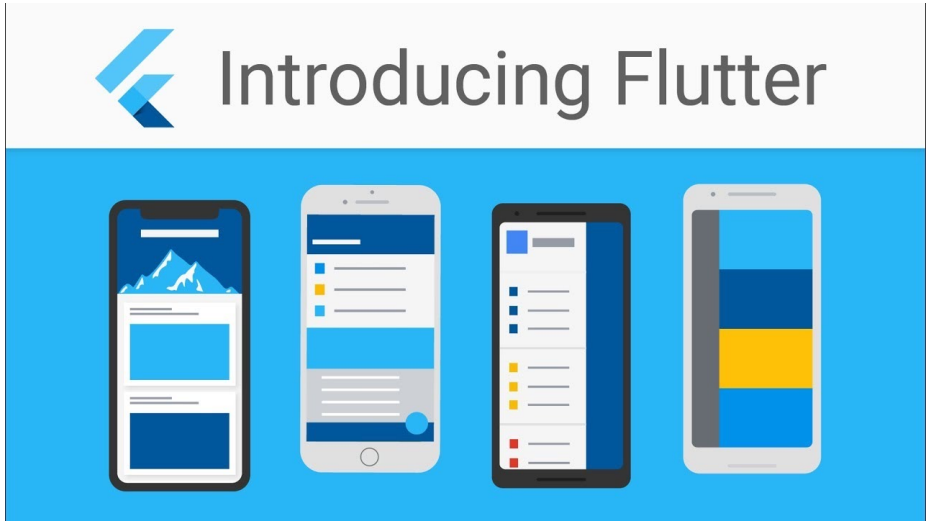
- **Нативное приложение** (native app) – прикладная программа, которая разработана для определенной платформы или для определенного устройства
- **Мобильное приложение** (mobile app) — прикладная программа, предназначенная для работы на смартфонах, планшетах и других мобильных (портативных, переносных, карманных) устройствах, собирается отдельно для iOS или Android
- **Десктопное приложение** (desktop app) — программа, которая устанавливается на компьютер пользователя и работает под управлением операционной системы, собирается отдельно для maxOS, Windows, Linux



Кроссплатформенная разработка

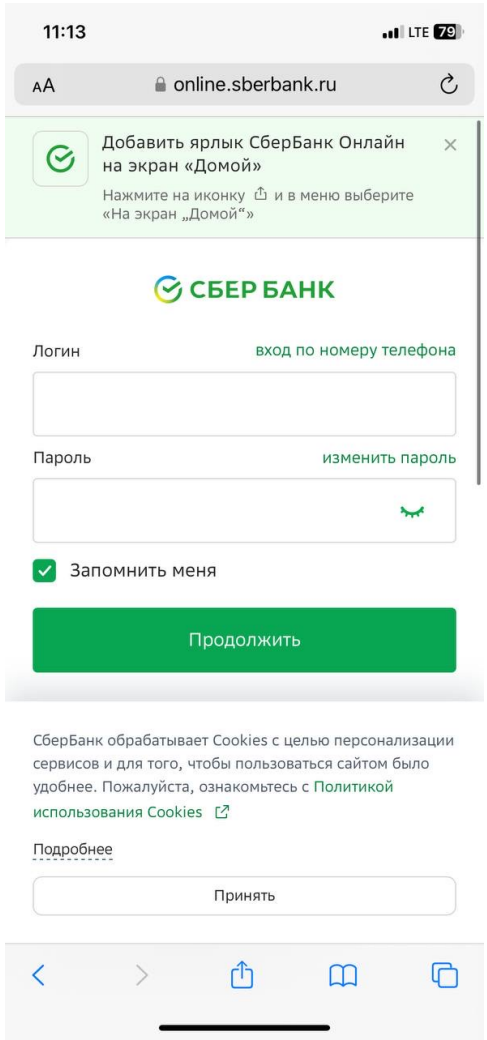
- Flutter, React Native, Kotlin multiplatform для кроссплатформенной мобильной разработки
- Flutter на Dart от Google
- React Native на JavaScript
- Kotlin multiplatform на Kotlin от JetBrains

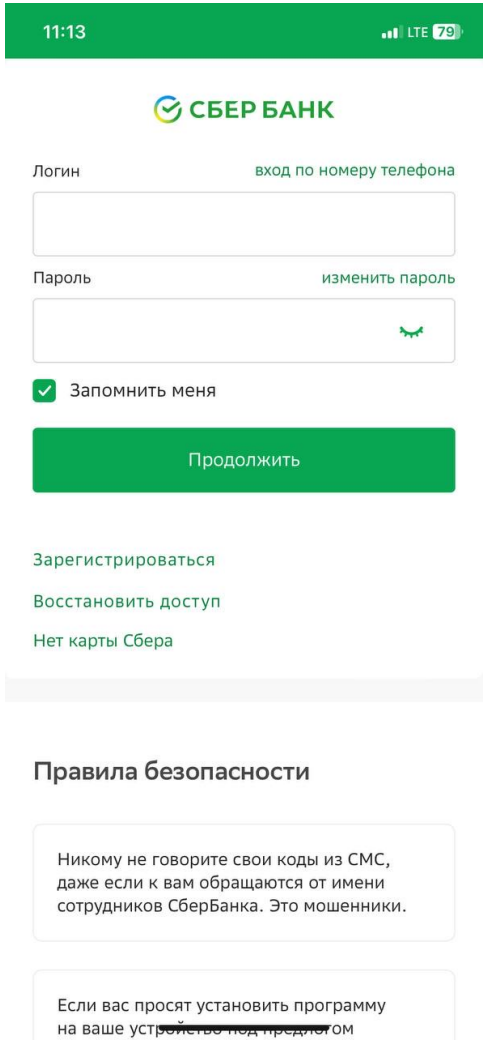


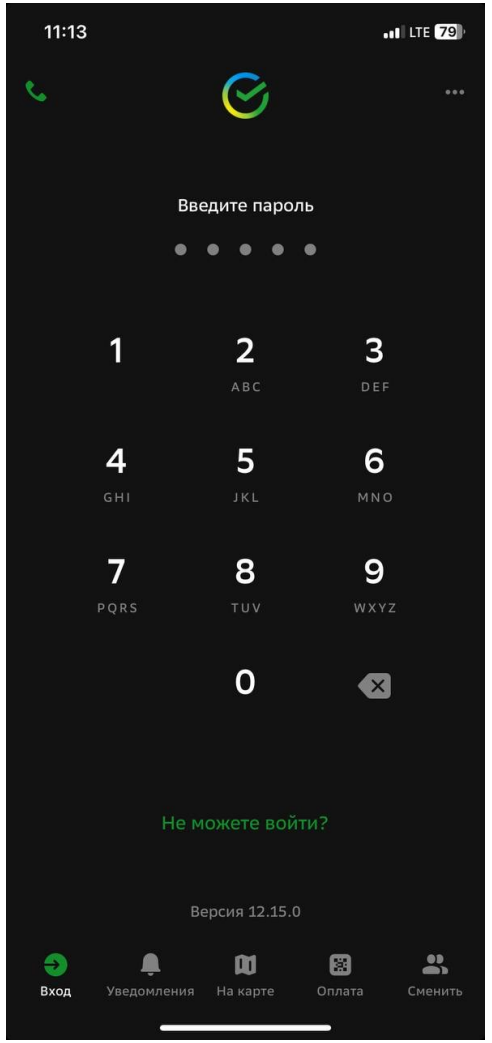


- Electron/Tauri и Qt для кроссплатформенных десктопных приложений
- Electron на JavaScript
- Tauri на JavaScript и Rust
- Qt на C++, PyQt на Python

Web vs мобильные приложения



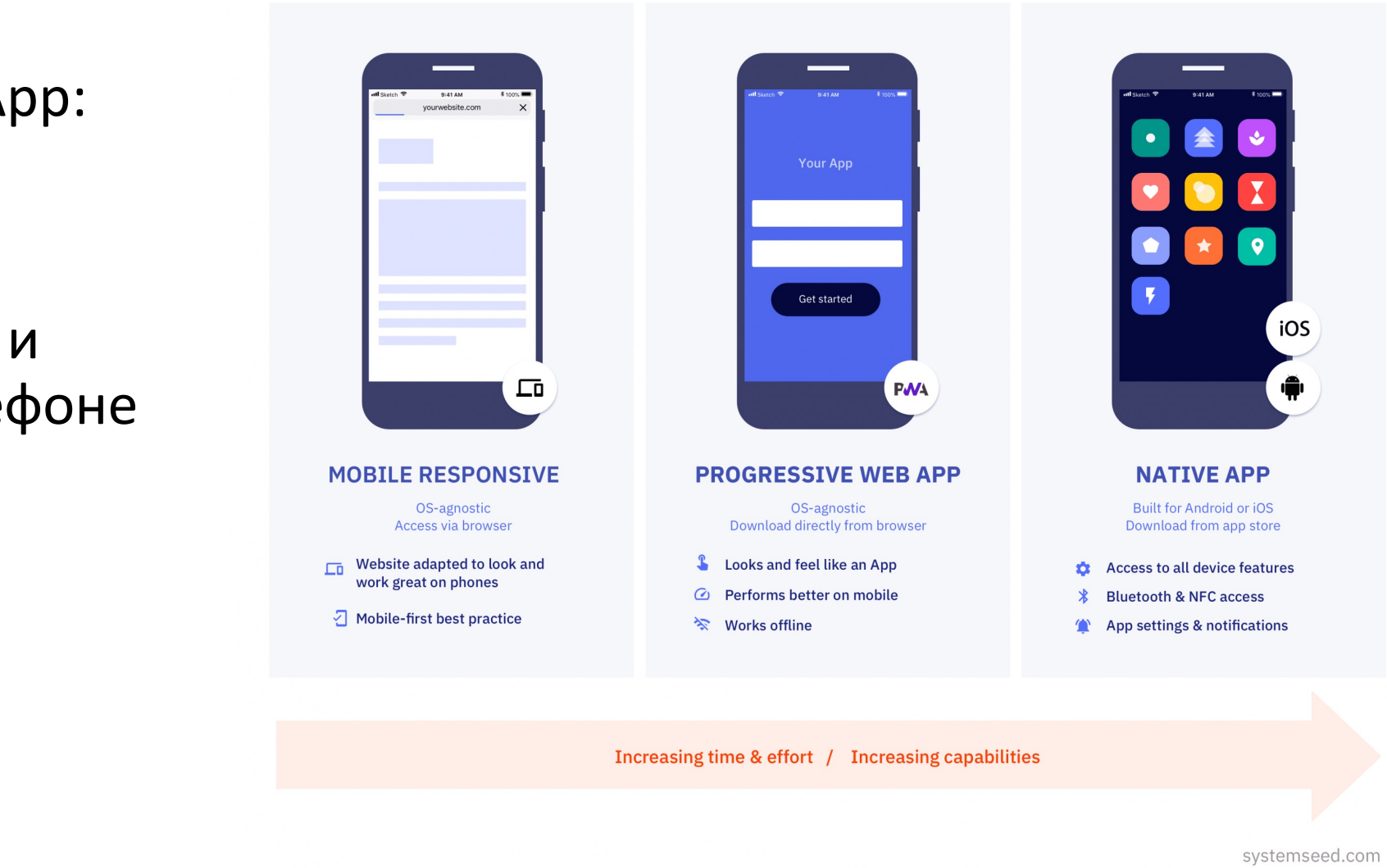




PWA

Progressive Web App:

- Выглядит как приложение
- Работает лучше и быстрее на телефоне
- Работает offline



PWA. manifest.json

```
{
  "name": "Tile Notes",
  "short_name": "Tile Notes",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#fdfdfd",
  "theme_color": "#db4938",
  "orientation": "portrait-primary",
  "icons": [
    {
      "src": "/logo192.png",
      "type": "image/png", "sizes": "192x192"
    },
    {
      "src": "/logo512.png",
      "type": "image/png", "sizes": "512x512"
    }
  ]
}
```

PWA. Service Worker

Регистрируем service worker, делаем это в файле `index.js` после рендера корневого компонента:

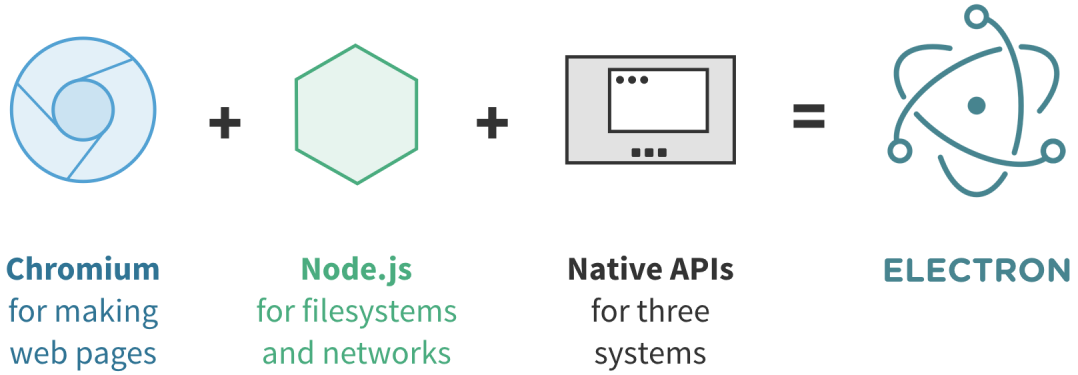
```
if ("serviceWorker" in navigator) {
  window.addEventListener("load", function() {
    navigator.serviceWorker
      .register("/serviceWorker.js")
      .then(res => console.log("service worker registered"))
      .catch(err => console.log("service worker not registered", err))
  })
}
```

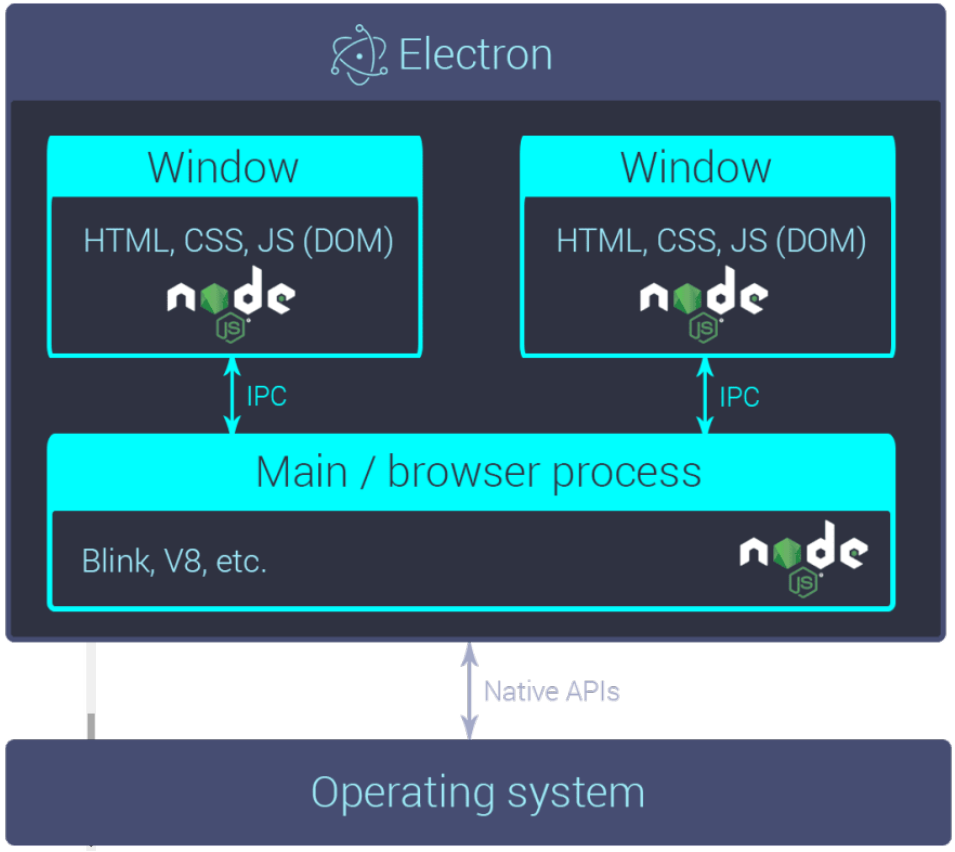
Создаем файл serviceWorker.js и кладем его в директорию public:

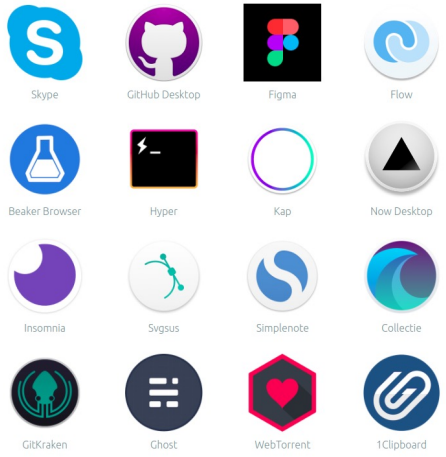
```
self.addEventListener('fetch',() => console.log("fetch"));
```

Electron

- **Electron** — фреймворк, разработанный GitHub.
- Позволяет разрабатывать нативные графические приложения для операционных систем с помощью веб-технологий, комбинируя возможности Node.js для работы с back-end и браузера Chromium



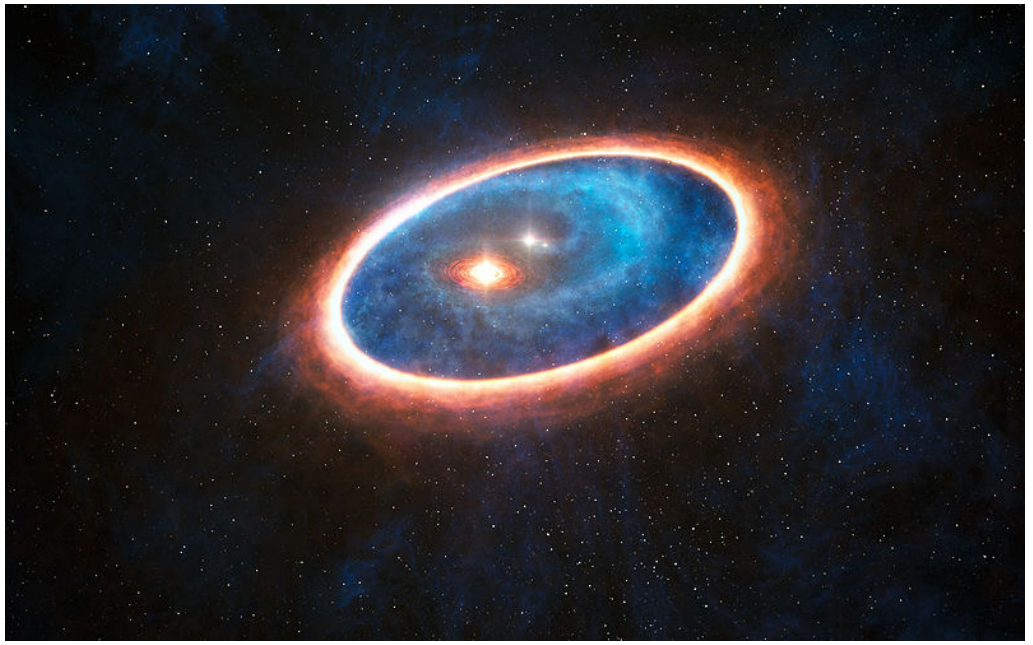


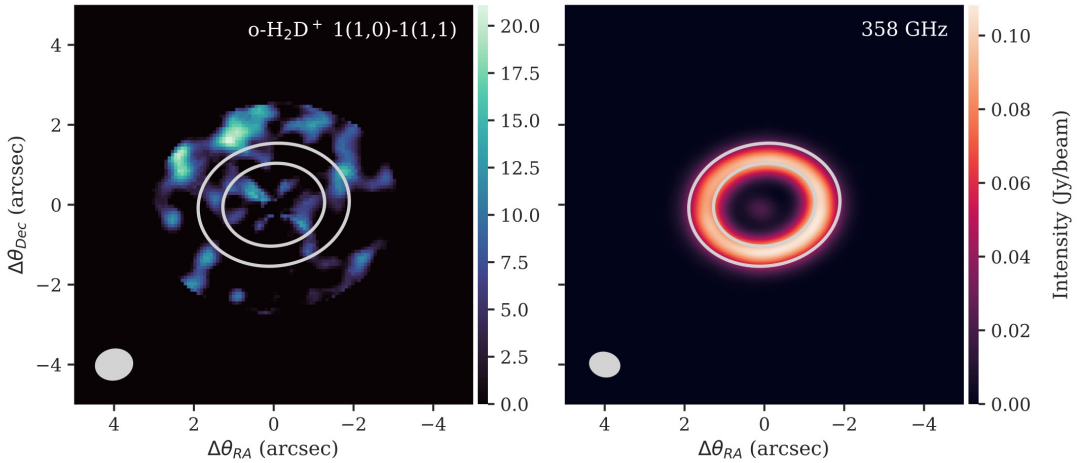


Tauri

- Фреймворк для создания десктопных приложений использующий Tao+Wry.
- Похожий на Electron, но позволяющий использовать Rust вместо Node.js, например, для взаимодействия с файловой системой.
- 2 октября 2024 вышла Tauri v2

https://tauri.app/blog/tauri-20/

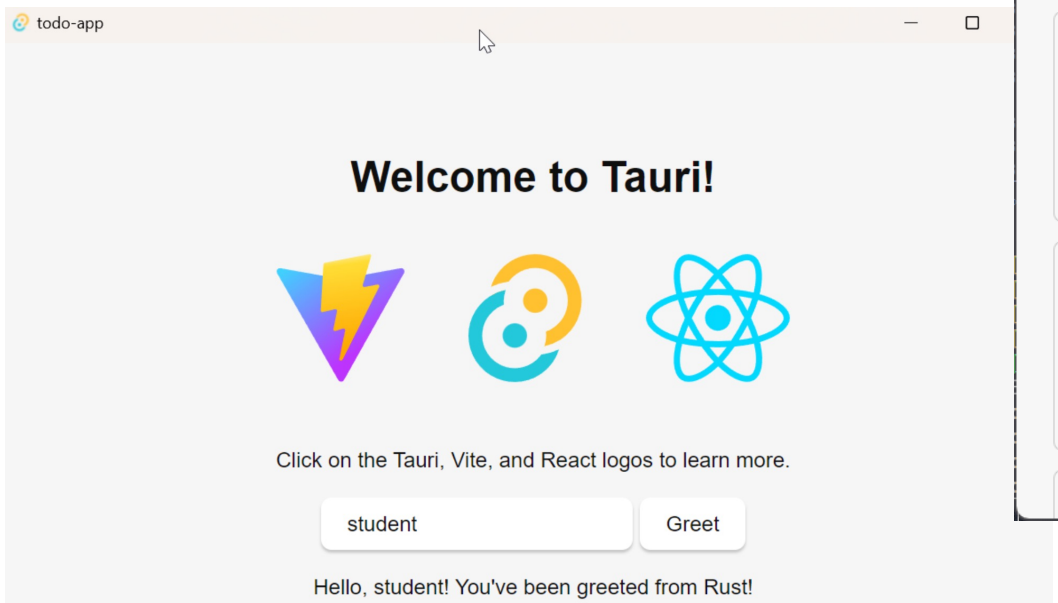


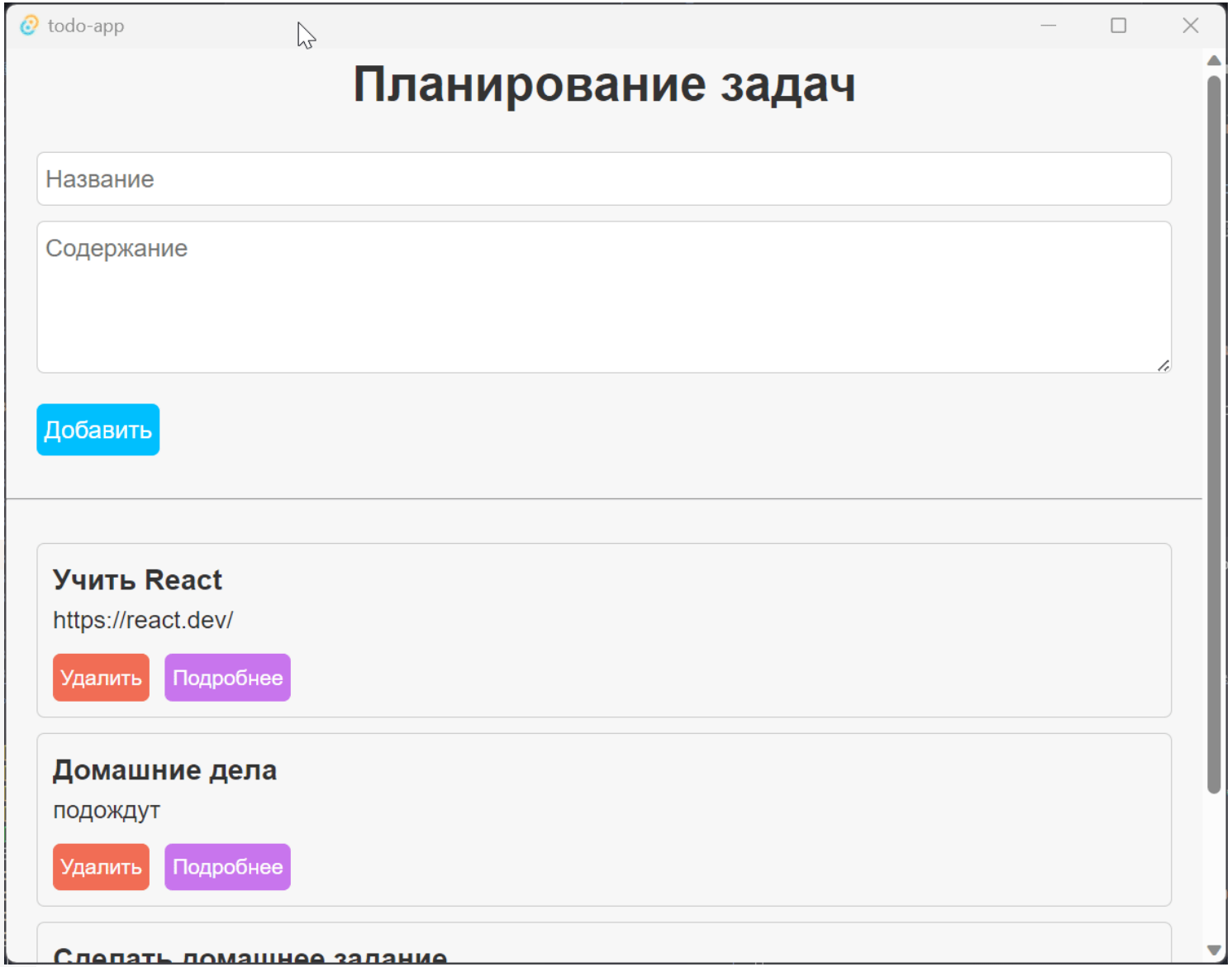


https://arxiv.org/pdf/2407.07238

Tauri

- Мы разрабатываем с помощью React приложение, которое имеет доступ к файловой системе, нативному меню, диалоговым окнам и к нашему API





Tauri

- Создание интерфейса Tauri приложения на React включает использование известных нам компонентов, обработчиков событий и хуков

```
const TodoListPage = () => {
    // список всех задач
    const [todos, setTodos] = useState([]);

    // новая задача
    const [newTodo, setNewTodo] = useState({ title: '', content: '' });

    // добавление новой задачи
    const handleAddTodo = () => {
        if (!newTodo.title || !newTodo.content) {
            console.error('Поля не должны быть пустыми');
            return;
        };
        const newTodoWithId = { ...newTodo, id: Date.now() };
        setTodos([...todos, newTodoWithId]);
        setNewTodo({ title: '', content: '' });
    };

    // удаление задачи
    const handleDeleteTodo = (id) => {
        const updatedTodos = todos.filter((todo) => todo.id !== id);
        setTodos(updatedTodos);
    };

    {/*return ( ... )*/}
}
```

```
export function TodoListPage() {
    // список всех задач
    const [todos, setTodos] = useState([]);

    // новая задача
    const [newTodo, setNewTodo] = useState({ title: '', content: '' });

    return (
        <div>
            <h1>Планирование задач</h1>
            <div className='container'>
                <input
                    className='input-title'
                    type='text'
                    placeholder='Название'
                    value={newTodo.title}
                />
                <textarea
                    className='input-content'
                    placeholder='Содержание'
                    value={newTodo.content}
                />
                <button className='button button-success text-lg'>
                  Добавить
                </button>
            </div>
            <hr />
            <div className='container'>
                {todos.map((todo) => (
                    <div className='todo' key={todo.id}>
                        <h3 className='todo-title'>
                            {todo.title}
                        </h3>
                        <p className='todo-content'>
                            {todo.content}
                        </p>
                        <button className='button button-danger text-md'>
                            Удалить
                        </button>
                    </div>
                ))}
            </div>
        </div>
    );
```

```
<button
    className='button button-success text-lg'
    onClick={handleAddTodo}
>
    Добавить
</button>
```

Dioxus

- Еще один фреймворк на Rust
- Нам интересен тем, что концепции React используются и в других языках тоже

```rust
use dioxus::prelude::*;

fn App(cx: Scope) -> Element {
    cx.render(rsx! {
        div {
            h1 { "Hello, Dioxus!" }
            p { "Welcome to my first Dioxus app." }
        }
    })
}
```

```rust
use dioxus::prelude::*;

fn Counter(cx: Scope) -> Element {
    let mut count = use_state(cx, || 0);
    cx.render(rsx! {
        div {
            h2 { "Counter: {count}" }
            button { onclick: move |_| count += 1, "Increment" }
        }
    })
}
```

https://best-of-web.builder.io/library/DioxusLabs/dioxus