ГУАП

КАФЕДРА № 42

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | С.Ю. Гуков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3  Системы контроля версий (VCS) |
| по курсу: Технологии программирования |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |
| --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4321 |  | Г.В. Буренков |
|  |  | подпись, дата | инициалы, фамилия |

Санкт-Петербург 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 2](#_Toc197104126)

[2 Задание 3](#_Toc197104127)

[3 Краткое описание хода работы и назначение технологий 4](#_Toc197104128)

[4 Ссылка на git с описанием авторства и назначения веток 5](#_Toc197104129)

[5 Скриншоты с сервиса git (ветки, коммиты, слияния) 7](#_Toc197104130)

[6 Скриншоты возникших конфликтов 10](#_Toc197104131)

[7 Результат работы приложения 11](#_Toc197104132)

[8 Вывод 13](#_Toc197104133)

[9 Листинг с кодом программы 14](#_Toc197104134)

**1 Цель работы**

Ознакомиться с функциональностью и принципами работы систем контроля версий, в частности Git. Изучить основные операции с репозиториями, принципы командной разработки и взаимодействия с Git-платформами. На практике реализовать командную работу над проектом, задействовав ветвление, pull requests, code review, слияние изменений и разрешение конфликтов.

**2 Задание**

Выполнить лабораторную работу в команде по 2-4 человека, используя платформу GitHub, создать репозиторий и присоединить остальных участников, разработать общий интерфейс программы, каждый участник должен выполнить свое задание по варианту, используя разные ветки, периодически делая коммиты и pull request в ветку dev, обновляя свой локальный проект кодом коллег, после выполнения всех заданий ветка dev сливается в главную ветку master, и оформляется файл README.md с пояснениями о выполненных заданиях..

**3 Краткое описание хода работы и назначение технологий**

Для реализации приложения по описанию задачи была выбрана модульная архитектура. Для реализации модульности по заданиям был реализован сайдбар, где были встроены автоматически генеративные разделы: главная, статистика по инфляции, миграции, пробежкам, рождаемости.

Были использованы технологии такие как:

* Javascript ⎯ мультипарадигменный язык программирования для реализации взаимодействия с веб-интерфейсом.
* Typescript ⎯ статический типизатор для языка Javascript. Позволяет обернуть данный язык в типы и статически проверять их.
* React ⎯ фреймворк на базе Javascript, который использует компонентный подход и позволяет динамически работать с созданными элементами DOM с помощью внутреннего механизма React Reconciliation.
* AntDesign ⎯ библиотека готовых дизайн компонентов на базе React. Позволяет быстро реализовывать приложения-макеты.
* Chart.js ⎯ пакет для реализации различных видов графиков.
* Vite ⎯ сборщик модулей для Javascript.
* Prettier ⎯ форматтер кода.
* Eslint ⎯ инструмент слежки за стайлгайдом проекта.
* Npm ⎯ сборщик пакетов для Javascript.

В рамках каждого задания соблюдены правила использованы пакеты, описанные выше, соблюдены правила eslint, код отформатирован по правилам prettier, модули имеют общую стилистикую

**4 Ссылка на git с описанием авторства и назначения веток**

Данная лабораторной работа храниться на платформе GitHub в публичном репозитории, где можно подробно отследить все действия за каждым из участников группы.

Ссылка на страницу проекта на GitHub: <https://github.com/lebedev05tmn/dashboards-guap>

Каждый из участников выполнял свое задание под вариантом и загружал выполненную версию в общий репозиторий в свою ветку:

lebedev05tmn (Лебедев Константин) - <https://github.com/lebedev05tmn>,

ветка feature/inflation, вариант задания номер 10 изображен на рисунке 1.

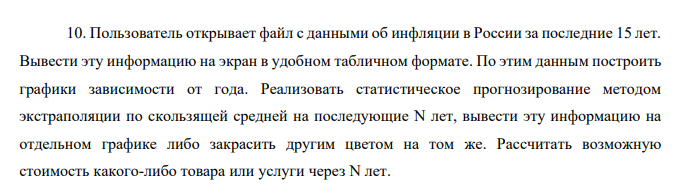


Рисунок 1 – Вариант номер 10

skv0r (Буренков Григорий) - <https://github.com/skv0r>,

ветка feature/jogging, вариант задания номер 1 изображен на рисунке 2.

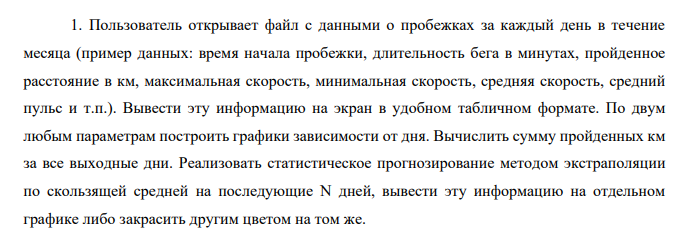


Рисунок 2 – Вариант номер 1

Qfimf (Пухловский Дмитрий) - <https://github.com/Qfimf>,

ветка feature/birthStatics, вариант задания номер 17 изображен на рисунке 3.

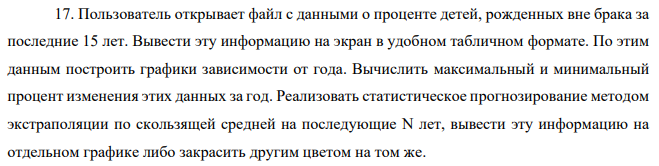


Рисунок 3 – Вариант номер 17

kosobutski (Илюхин Кирилл) - <https://github.com/kosobutski>,

ветка feature/migration-statisctics, вариант задания номер 6 изображен на рисунке 3.

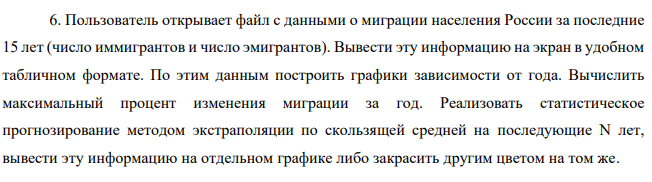


Рисунок 4 – Вариант номер 6

**5 Скриншоты с сервиса git (ветки, коммиты, слияния)**

Во время выполнения работы каждый из участников выполнял скриншоты своих коммитов, веток, пулреквестов.На рисунках 4-8 предоставлены скриншоты с сервиса GitHub, демонстрирующие создание репозитория, создание веток, коммиты, pull request и слияния веток и код ревью чужих веток.

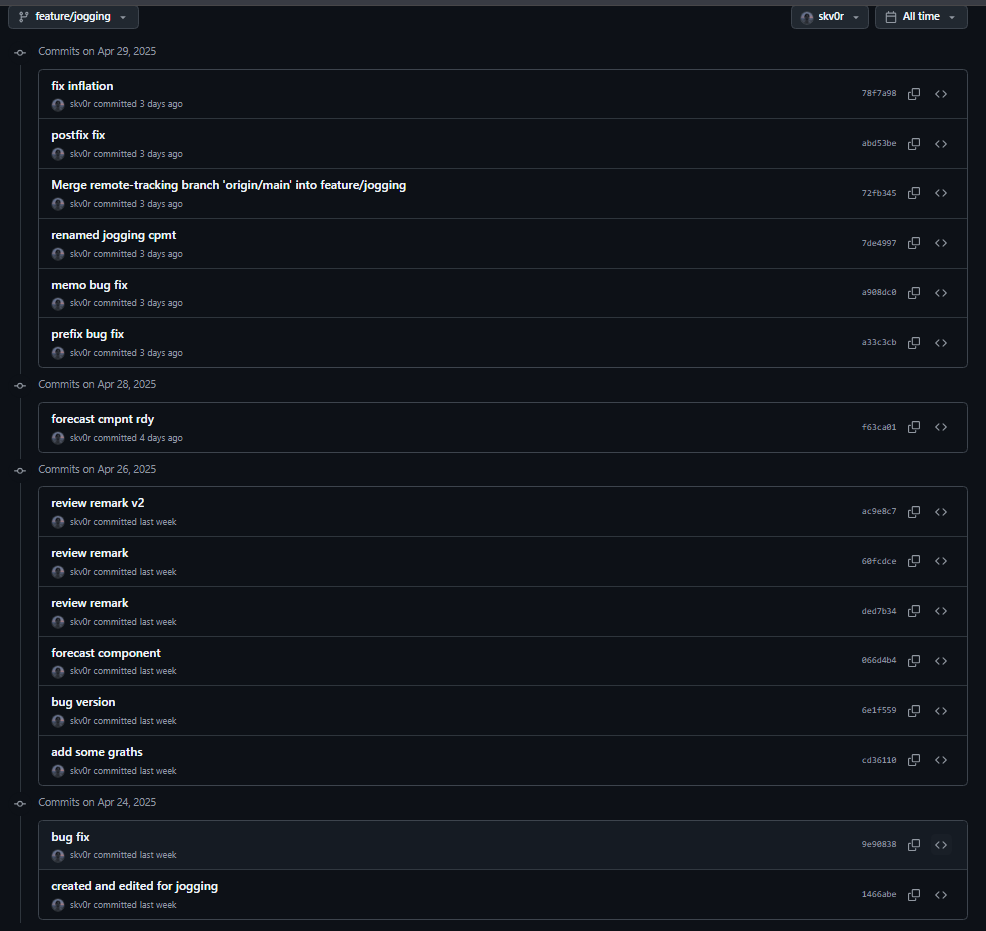


Рисунок 5 – Скриншот коммитов в ветке feature/jogging

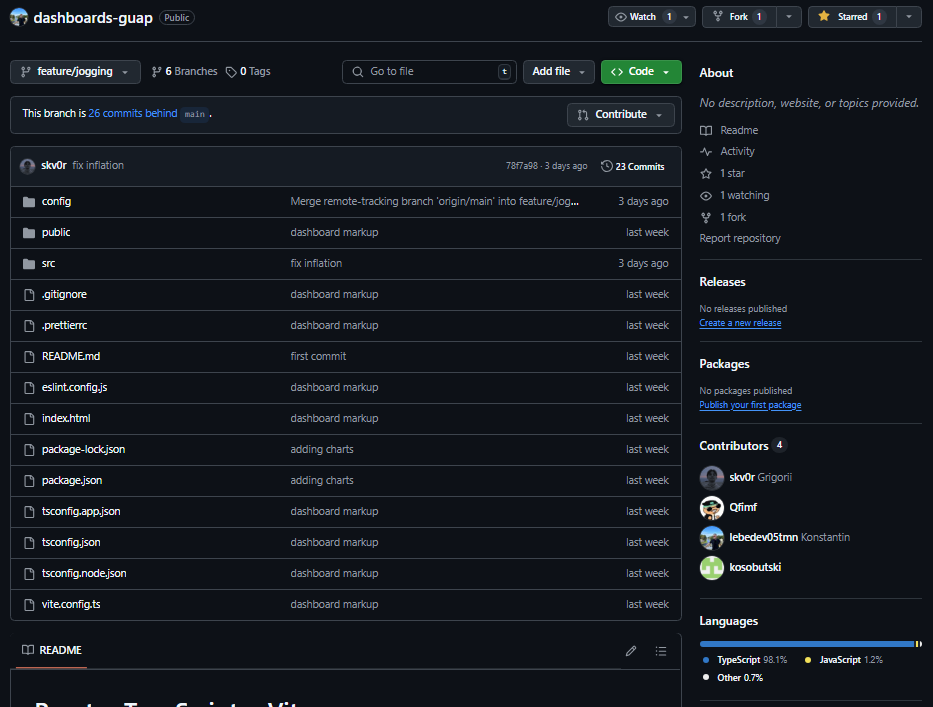


Рисунок 6 – Ветка feature/jogging

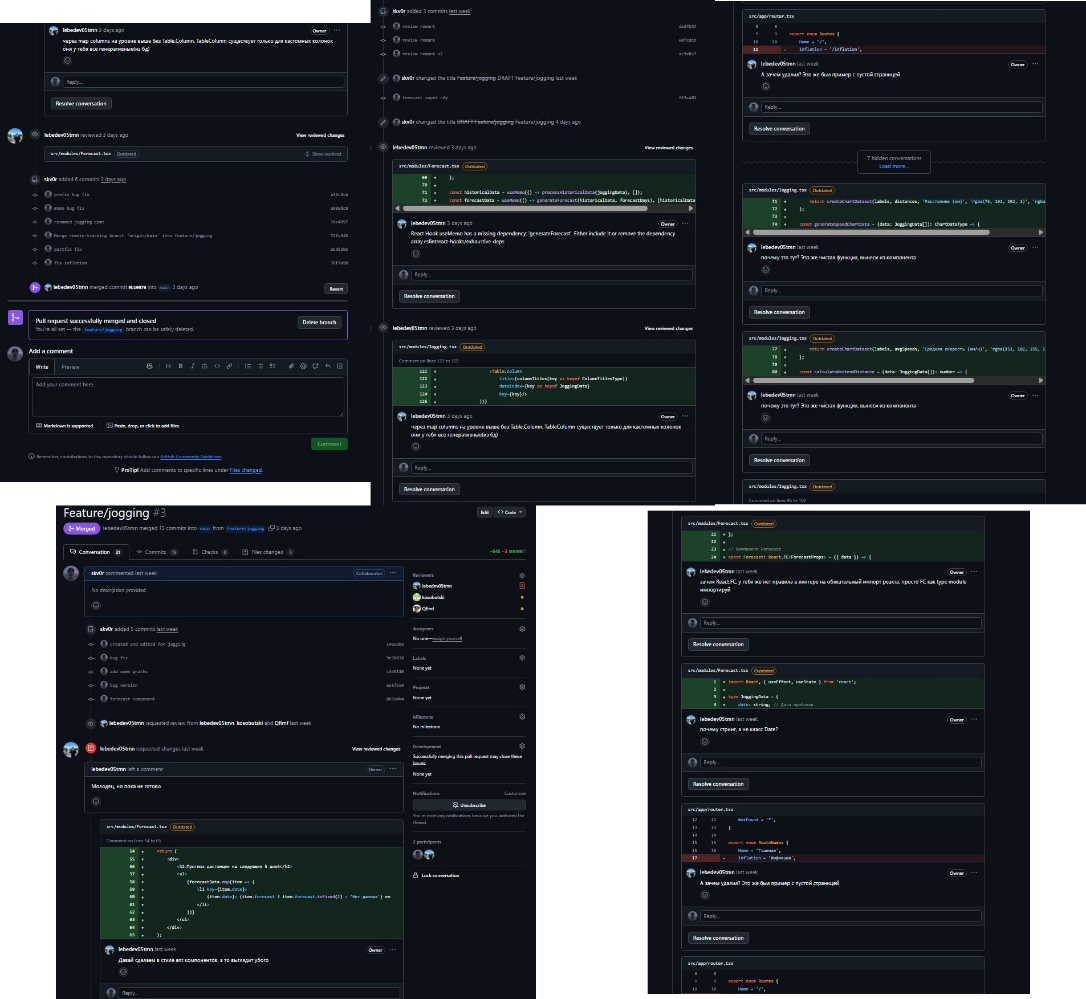


Рисунок 7 – Code review и Pull Request в Ветке feature/jogging

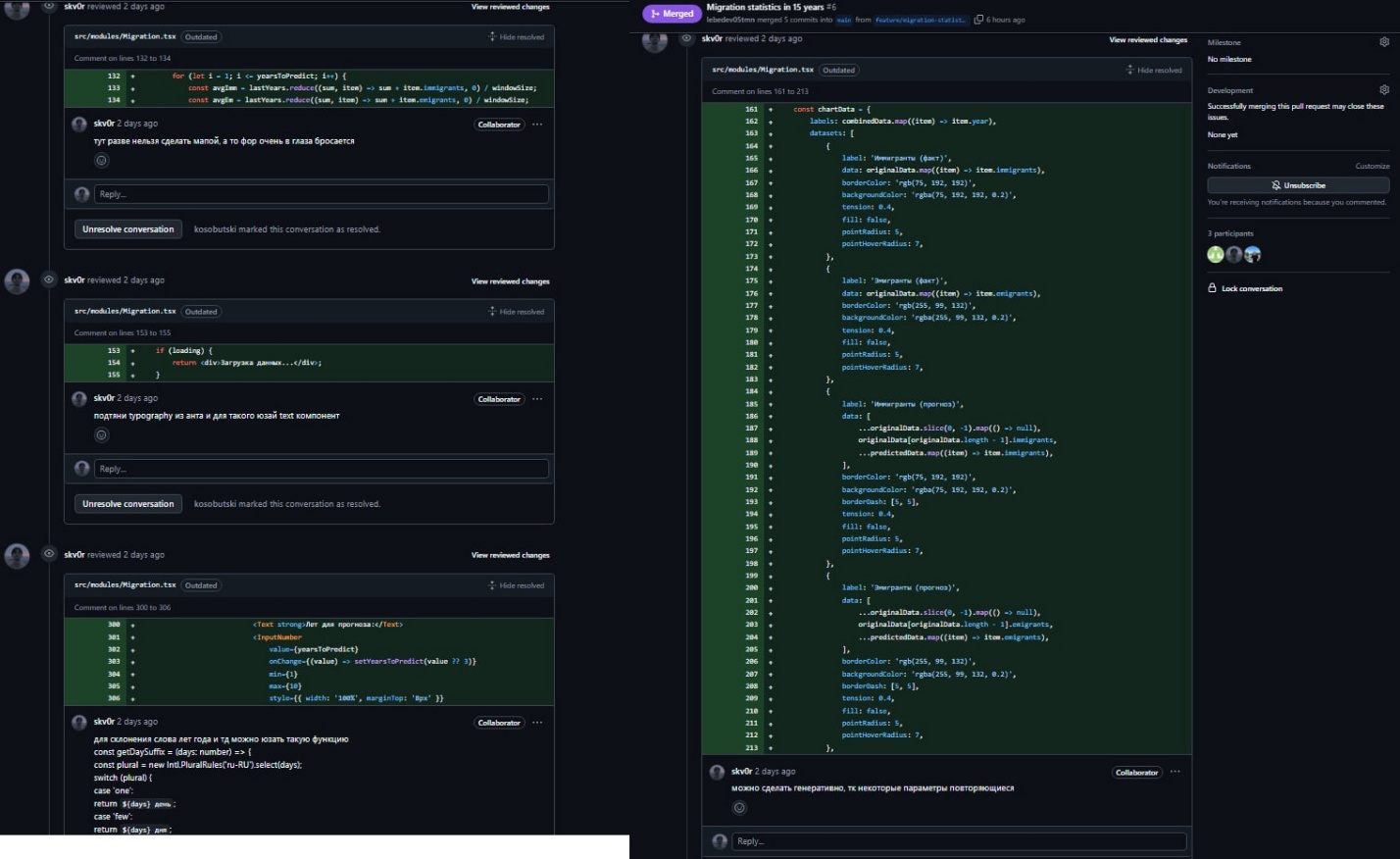


Рисунок 8 – Мой Code review в Ветке feature/birthStatics

**6 Скриншоты возникших конфликтов**

Во время разработки я столкнулся с конфликтом во время слияния своей ветки feature/jogging с веткой main. На рисунках 9-10 изображены возникшие конфликты.

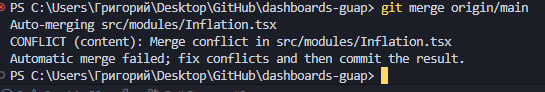


Рисунок 9 – merge conflict во время слияния веток

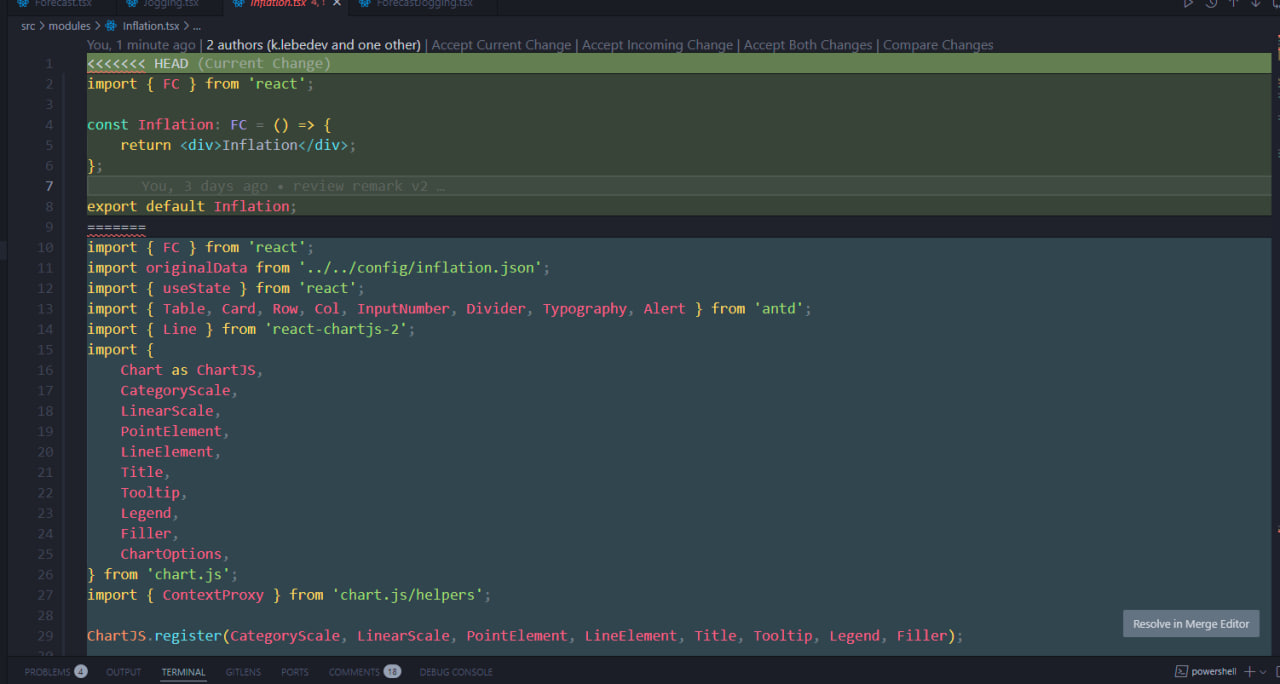


Рисунок 10 – конфликт в файле Inflation.tsx

**7 Результат работы приложения**

Результаты работы программы демонстрируют результаты работы программы, демонстрирующие различные сценарии использования, включая вывод данных на экран, построение графиков и выполнение вычислений. Данный даэшборд легко расширяемый и может быть использован для любых задач.

На рисунке 11 изображена главная страница дэшборда Home.

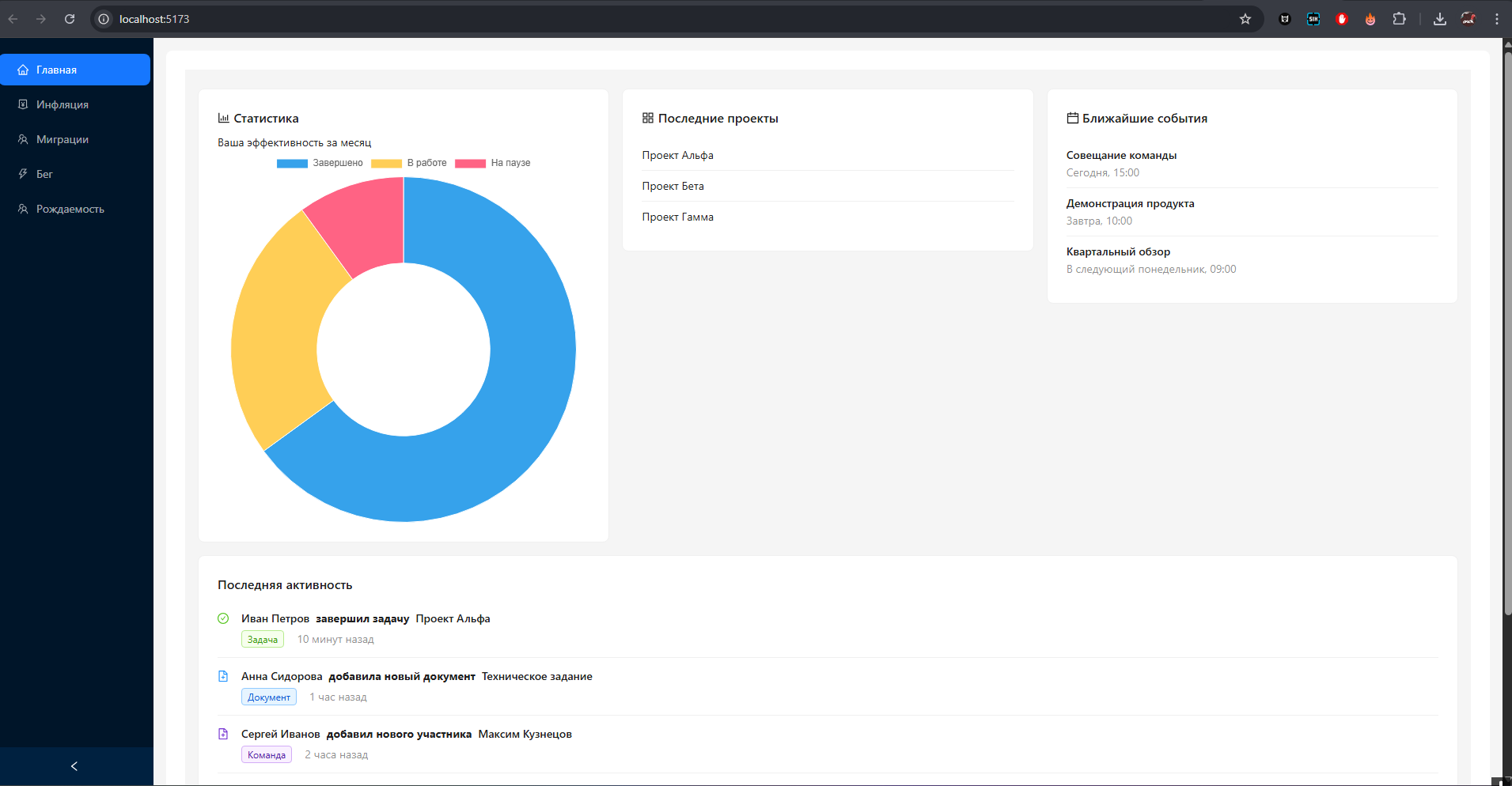
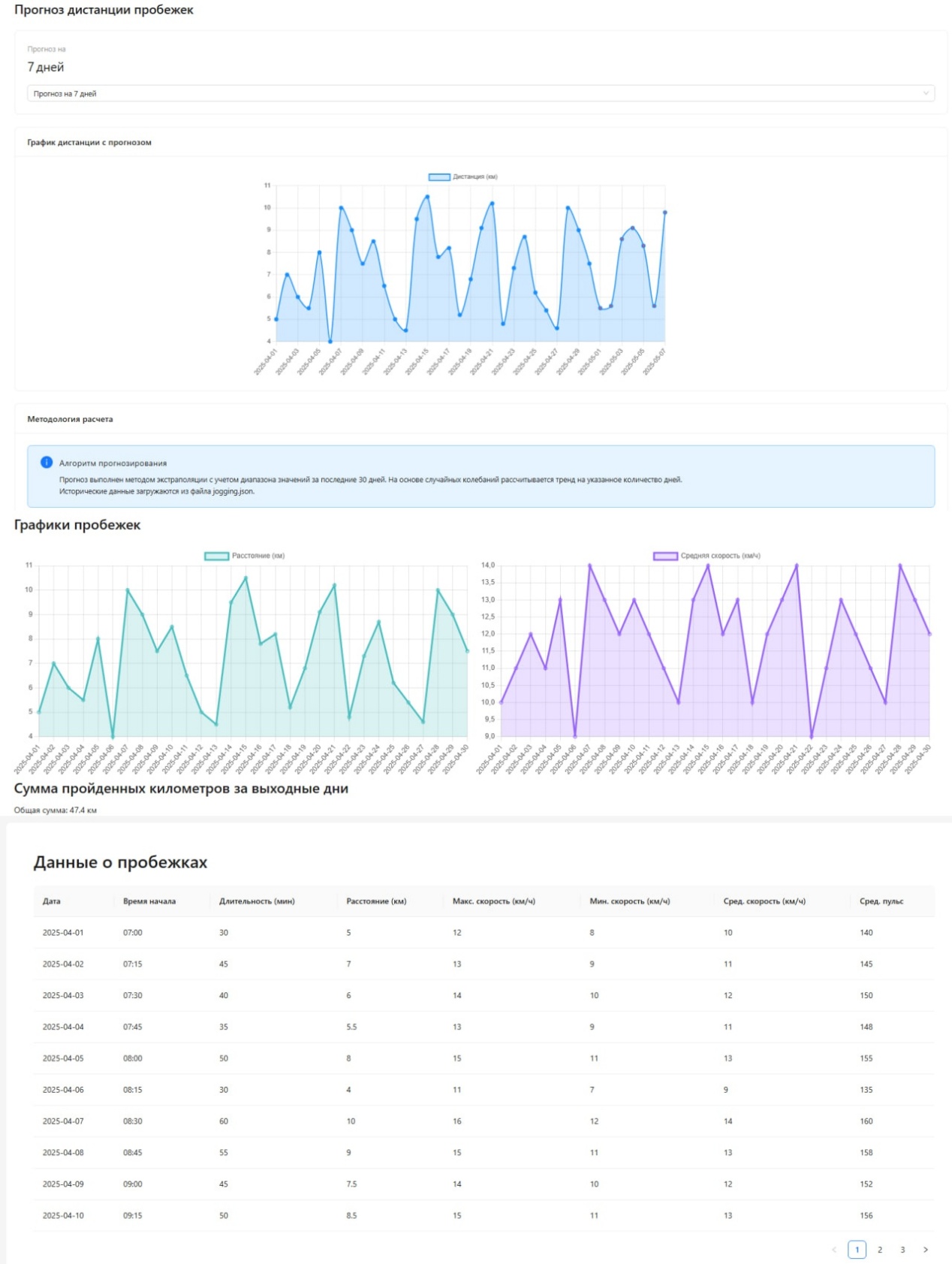


Рисунок 11 – Результат работы приложения



На рисунке 12 представлен результат работы страницы Jogging (Бег).

**8 Вывод**

В рамках данной работы все участники команды познакомились с работой в git, получили важнейший опыт работы в команде: сливание веток, ревью, параллельное программирование. Научись: создавать ветки, пулл реквесты, исправлять свои ошибки, решать конфликты, поддерживать стайлгайд.

В выводе можно отметить, что git ⎯ одна из самых удобнейших систем контроля версий, платформа github ⎯ хорошее бесплатное решение для работы в команде, в данный момент не представляется возможным хорошая и удобная совместная работа над проектом без использования каких-либо VCS.

**9 Листинг с кодом программы**

В данном разделе представлен исходный код моей части программы:

//TYPESCRIPT

import React, { useEffect, useState } from 'react';

import originalData from '../../config/jogging.json';

import { Table, Typography } from 'antd';

import { Line } from 'react-chartjs-2';

import {

Chart as ChartJS,

LineElement,

PointElement,

LinearScale,

Title,

Tooltip,

Legend,

CategoryScale,

} from 'chart.js';

import Forecast from '../ui/ForecastJogging';

ChartJS.register(LineElement, PointElement, LinearScale, Title, Tooltip, Legend, CategoryScale);

const { Title: AntTitle, Text } = Typography;

type JoggingData = {

date: string;

startTime: string;

duration: number;

distance: number;

maxSpeed: number;

minSpeed: number;

avgSpeed: number;

avgPulse: number;

};

type ChartDataType = {

labels: string[];

datasets: {

label: string;

data: number[];

borderColor: string;

backgroundColor: string;

fill: boolean;

}[];

};

type StateType = {

data: JoggingData[];

distanceChartData: ChartDataType | null;

speedChartData: ChartDataType | null;

totalWeekendDistance: number;

};

const columnTitles: { [key in keyof JoggingData]: string } = {

date: 'Дата',

startTime: 'Время начала',

duration: 'Длительность (мин)',

distance: 'Расстояние (км)',

maxSpeed: 'Макс. скорость (км/ч)',

minSpeed: 'Мин. скорость (км/ч)',

avgSpeed: 'Сред. скорость (км/ч)',

avgPulse: 'Сред. пульс',

};

const createChartDataset = (

labels: string[],

distances: number[],

label: string,

borderColor: string,

backgroundColor: string,

): ChartDataType => {

return {

labels,

datasets: [

{

label,

data: distances,

borderColor,

backgroundColor,

fill: true,

},

],

};

};

const processData = (data: JoggingData[]): JoggingData[] => {

return data.map((item) => ({

key: item.date,

...item,

}));

};

const generateDistanceChartData = (data: JoggingData[]): ChartDataType => {

const labels = data.map((item) => item.date);

const distances = data.map((item) => item.distance);

return createChartDataset(labels, distances, 'Расстояние (км)', 'rgba(75, 192, 192, 1)', 'rgba(75, 192, 192, 0.2)');

};

const generateSpeedChartData = (data: JoggingData[]): ChartDataType => {

const labels = data.map((item) => item.date);

const avgSpeeds = data.map((item) => item.avgSpeed);

return createChartDataset(

labels,

avgSpeeds,

'Средняя скорость (км/ч)',

'rgba(153, 102, 255, 1)',

'rgba(153, 102, 255, 0.2)',

);

};

const calculateWeekendDistance = (data: JoggingData[]): number => {

return data.reduce((total, item) => {

const date = new Date(item.date);

const day = date.getDay(); // 0 - воскресенье, 6 - суббота

if (day === 0 || day === 6) {

return total + item.distance;

}

return total;

}, 0);

};

const columns = Object.keys(columnTitles).map((key) => ({

title: columnTitles[key as keyof JoggingData],

dataIndex: key as keyof JoggingData,

key: key,

}));

const Jogging: React.FC = () => {

const [chartData, setChartData] = useState<StateType>({

data: [],

distanceChartData: null,

speedChartData: null,

totalWeekendDistance: 0,

});

useEffect(() => {

const processedData = processData(originalData);

setChartData({

data: processedData,

distanceChartData: generateDistanceChartData(processedData),

speedChartData: generateSpeedChartData(processedData),

totalWeekendDistance: calculateWeekendDistance(processedData),

});

}, []);

if (!chartData) return null;

return (

< div style={{ padding: '24px' }}>

<AntTitle level={2} style={{ marginBottom: '24px' }}>

Данные о пробежках

</AntTitle>

<Table dataSource={chartData.data} pagination={{ pageSize: 10 }} columns={columns} rowKey="date"/>

<AntTitle level={3} style={{ marginBottom: '24px' }}>

Графики пробежек

</AntTitle>

<div style={{ display: 'flex', justifyContent: 'space-between' }}>

<div style={{ width: '50%' }}>

{chartData.distanceChartData ? (

<Line data={chartData.distanceChartData} />

) : (

<Text>Загрузка данных для графика дистанции...</Text>

)}

</div>

<div style={{ width: '50%' }}>

{chartData.speedChartData ? (

<Line data={chartData.speedChartData} />

) : (

<Text>Загрузка данных для графика скорости...</Text>

)}

</div>

</div>

<AntTitle level={3} style={{ marginBottom: '12px' }}>

Сумма пройденных километров за выходные дни

</AntTitle>

<Text style={{ marginBottom: '24px' }}>{`Общая сумма: ${chartData.totalWeekendDistance} км`}</Text>

<Forecast />

</div>

);

};

export default Jogging;

#ui component forecastjogging.tsx

import React, { useState } from 'react';

import { Card, Statistic, Select, Typography, Alert } from 'antd';

import { Line } from 'react-chartjs-2';

import { Chart as ChartJS, CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend, Filler } from 'chart.js';

import joggingData from '../../config/jogging.json';

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend, Filler);

const { Title: AntTitle, Text } = Typography;

const DAYS\_FOR\_FORECAST = 7;

const RANDOM\_CHANGE\_PERCENTAGE = 0.1;

const getDaySuffix = (days: number) => {

const plural = new Intl.PluralRules('ru-RU').select(days);

switch (plural) {

case 'one':

return `${days} день`;

case 'few':

return `${days} дня`;

default:

return `${days} дней`;

}

};

interface JoggingData {

date: string;

startTime: string;

duration: number;

distance: number;

maxSpeed: number;

minSpeed: number;

avgSpeed: number;

avgPulse: number;

}

interface ProcessedData extends JoggingData {

isForecast?: boolean;

}

const processHistoricalData = (data: JoggingData[]): ProcessedData[] => {

return data.map((item) => ({

...item,

isForecast: false

}));

};

const calculateForecastDistance = (avgDistance: number, minDistance: number, maxDistance: number): number => {

const randomFactor = (Math.random() \* (RANDOM\_CHANGE\_PERCENTAGE \* 7) - RANDOM\_CHANGE\_PERCENTAGE \* 3) \* avgDistance;

return Math.max(minDistance, Math.min(maxDistance, avgDistance + randomFactor));

};

const generateForecast = (data: ProcessedData[], days: number): ProcessedData[] => {

const lastDistances = data.slice(-30).map(item => item.distance);

const minDistance = Math.min(...lastDistances);

const maxDistance = Math.max(...lastDistances);

const avgDistance = lastDistances.reduce((sum, value) => sum + value, 0) / lastDistances.length;

const lastDate = new Date(data[data.length - 1].date);

return Array.from({ length: days }, (\_, i) => {

const newDate = new Date(lastDate);

newDate.setDate(lastDate.getDate() + i + 1);

const predictedDistance = calculateForecastDistance(avgDistance, minDistance, maxDistance);

return {

date: newDate.toISOString().split('T')[0],

startTime: "00:00",

duration: 0,

distance: Number(predictedDistance.toFixed(1)),

maxSpeed: 0,

minSpeed: 0,

avgSpeed: 0,

avgPulse: 0,

isForecast: true

};

});

};

const ForecastJogging: React.FC = () => {

const [forecastDays, setForecastDays] = useState<number>(DAYS\_FOR\_FORECAST);

const historicalData = processHistoricalData(joggingData);

const forecastData = generateForecast(historicalData, forecastDays);

const allData = [...historicalData, ...forecastData];

const chartData = {

labels: allData.map(item => item.date),

datasets: [

{

label: 'Дистанция (км)',

data: allData.map(item => item.distance),

borderColor: '#1890ff',

backgroundColor: 'rgba(24, 144, 255, 0.2)',

borderWidth: 2,

pointBackgroundColor: allData.map(item => item.isForecast ? '#ff4d4f' : '#1890ff'),

tension: 0.3,

fill: true

}

]

};

return (

<div>

<AntTitle level={3} style={{ marginBottom: '24px' }}>

Прогноз дистанции пробежек

</AntTitle>

<Card style={{ marginBottom: '24px' }}>

<div style={{ marginBottom: '16px' }}>

<Statistic title="Прогноз на" value={forecastDays} suffix={forecastDays === 1 ? 'день' : 'дней'} />

</div>

<Select

style={{ width: '100%' }}

value={forecastDays}

onChange={setForecastDays}

options={[1, 3, 5 , 7, 10].map(d => ({

value: d,

label: `Прогноз на ${getDaySuffix(d)}`

}))}

/>

</Card>

<Card title="График дистанции с прогнозом" style={{ marginBottom: '24px' }}>

<div style={{ width: '50%', height: '400px', margin: 'auto'}}>

<Line data={chartData} options={{ responsive: true }} />

</div>

</Card>

<Card title="Методология расчета">

<Alert

message="Алгоритм прогнозирования"

description={

<>

<Text>

Прогноз выполнен методом экстраполяции с учетом диапазона значений за последние 30 дней.

На основе случайных колебаний рассчитывается тренд на указанное количество дней.

</Text>

<br />

<Text>

Исторические данные загружаются из файла jogging.json.

</Text>

</>

}

type="info"

showIcon

/>

</Card>

</div>

);

};

export default ForecastJogging;