САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 3: Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS

Выполнила:
Едигарева Дарья
Группа К3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

Задача

Мигрировать ранее разработанное приложение (в рамках ЛР1 и ЛР2) на фреймворк Vue.JS.

Каким требованиям ваше приложение должно соответствовать:

Должен быть подключён роутер

Должна быть реализована работа с внешним API (желательно посредством axios)

Разумное деление на компоненты (продемонстрируйте понимание компонентного подхода)

Использование composable для выделения повторяющеся функционала в отдельные файлы

HTML-точка входа

```
<!DOCTYPE html>

<html lang="ru">

 <head>

   <meta charset="UTF-8">

   <link rel="icon" href="/favicon.ico">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>JobBridge - поиск работы</title>

   <link
href="https://fonts.googleapis.com/css2?family=Manrope:wght@400;600;700&display=swa
p" rel="stylesheet">

 </head>

 <body class="page">

   <div id="app"></div>

   <script type="module" src="/src/main.js"></script>

 </body>
```

```
</html>
```

 - корневой контейнер, куда Vue монтирует приложение.

Подключение main.js как точки входа.

Инициализация Vue-приложения

main.js

```
import { createApp } from 'vue'

import { createPinia } from 'pinia'



import App from './App.vue'

import router from './router'



import 'bootstrap/dist/css/bootstrap.min.css'

import 'bootstrap'

import '@/assets/styles.css'



const app = createApp(App)



app.use(createPinia())

app.use(router)



app.mount('#app')
```

createApp(App) создаёт приложение.

createPinia() подключает хранилище.

router подключает маршрутизацию.

Bootstrap CSS/JS + кастомные стили подключаются глобально.

Корневой компонент

```
<template>
  <main-layout>
    <router-view />
  </main-layout>
</template>
<script>
import MainLayout from '@/layouts/MainLayout.vue'
export default {
  name: 'App',
  components: { MainLayout },
}
</script>
```

MainLayout задаёт шапку и футер.

<router-view /> - место, где отображаются страницы в зависимости от маршрута.

Router

```
import { createRouter, createWebHistory } from 'vue-router'
const routes = [
  { path: '/', name: 'home', component: () => import('@/views/HomeView.vue') },
  { path: '/jobs', name: 'jobs', component: () => import('@/views/JobsView.vue') },
```

{ path: '/vacancies/:id', name: 'vacancy', component: () => import('@/views/VacancyView.vue'), props: true },

{ path: '/favorites', name: 'favorites', component: () => import('@/views/FavoritesView.vue') },

{ path: '/profile', name: 'profile', component: () => import('@/views/ProfileView.vue') },

{ path: '/employer', name: 'employer', component: () => import('@/views/EmployerView.vue') },

]

export default createRouter({

history: createWebHistory(import.meta.env.BASE_URL),

routes,

})

Используется createWebHistory - чистые URL без #.

Маршруты ведут на HomeView, JobsView, VacancyView, FavoritesView, ProfileView, EmployerView. SPA-навигация без перезагрузки.

API-слой

instance.js

```
import axios from 'axios'



const apiURL = 'http://localhost:3001'



const instance = axios.create({

 baseURL: apiURL,

})
```

```
export default instance
```

axios.create создаёт общий инстанс с baseURL для всех запросов.

Это единая точка конфигурации API.
jobs.js

```
export default (api) => ({
  getCompanies: () => api.get('/companies'),
  getVacancies: () => api.get('/vacancies'),
  getUsers: () => api.get('/users'),
  getProfiles: (userId) => api.get('/profiles', { params: userId ? { userId } : {} }),
  createProfile: (data) => api.post('/profiles', data),
  updateProfile: (id, data) => api.put(`/profiles/${id}`, data),
  getEmployers: (userId) => api.get('/employers', { params: { userId } }),
  createEmployer: (data) => api.post('/employers', data),
  createVacancy: (data) => api.post('/vacancies', data),
  getResponses: (userId) => api.get('/responses', { params: userId ? { userId } :
{} }),
  createResponse: (data) => api.post('/responses', data),
  login: (data) => api.post('/login', data),
  register: (data) => api.post('/register', data),
})
```

Класс JobsApi инкапсулирует все запросы: вакансии, компании, профили, отклики, логин/регистрация.

Каждый метод возвращает promise axios.

```
import instance from '@/api/instance'

import JobsApi from '@/api/jobs'



const jobsApi = new JobsApi(instance)



export { jobsApi }
```

Экспортируется один объект jobsApi, который используется в Pinia-store.

Pinia-store

jobs.js

state: хранит данные приложения: вакансии, компании, профили, избранное, отклики, текущий пользователь.

getters:

companiesMap и usersMap ускоряют доступ по id.

normalizedVacancies нормализует вакансии (добавляет лейблы и теги).

appliedVacancyIds используется для проверки, был ли отклик.

actions:

loadCompanies, loadVacancies, loadUsers, loadResponses, loadProfile - загрузка данных из API.

login, register, logout - авторизация через json-server-auth.

toggleFavorite - избранное сохраняется в localStorage.

applyToVacancy - создаёт отклик и сохраняет в state.

Константы и утилиты

industries.js

```
export const INDUSTRIES = [

 { value: 'technology', label: 'Технологии' },
```

```
 { value: 'finance', label: 'Финансы' },
 { value: 'healthcare', label: 'Здравоохранение' },
 { value: 'education', label: 'Образование' },
 { value: 'manufacturing', label: 'Производство' },
 { value: 'other', label: 'Другое' },
]


export const INDUSTRY_LABELS = INDUSTRIES.reduce(
 (acc, item) => ({ ...acc, [item.value]: item.label }),
 {},
)
```

Хранится список отраслей.

INDUSTRY_LABELS используется для отображения понятных названий.

labels.js

```
export const toExperienceBucket = (value) => {
 const num = Number(value) || 0
 if (num <= 0) return 'no'
 if (num <= 3) return '1-3'
 if (num <= 6) return '3-6'
 return '6+'
}
export const experienceLabel = (bucket) =>
 (
   {
     no: 'Нет опыта',
```

```
    '1-3': '1-3 года',
    '3-6': '3-6 лет',
    '6+': '6+ лет',
  }[bucket] || 'Любой опыт'
)
export const formatLabel = (value) =>
(
  {
    hybrid: 'Гибрид',
    remote: 'Удалённо',
    office: 'Офис',
  }[value] || 'Не указано'
)
```

Функции для преобразования опыта и форматов в читаемые лейблы.

Composables

useTheme.js

```
import { computed, onMounted, ref, watch } from 'vue'
const THEME_KEY = 'theme'
export const useTheme = () => {
 const theme = ref('light')
 const applyTheme = (value) => {
   const next = value === 'dark' ? 'dark' : 'light'
   document.documentElement.setAttribute('data-theme', next)
   localStorage.setItem(THEME_KEY, next)
   theme.value = next
```

```
}

const toggleTheme = () => {

  applyTheme(theme.value === 'dark' ? 'light' : 'dark')

}

onMounted(() => {

  const saved = localStorage.getItem(THEME_KEY)

  const prefersDark =

    window.matchMedia &&

    window.matchMedia('(prefers-color-scheme: dark)').matches

  applyTheme(saved || (prefersDark ? 'dark' : 'light'))

})

watch(theme, (value) => {

  document.documentElement.setAttribute('data-theme', value)

})



return {

  theme,

  isDark: computed(() => theme.value === 'dark'),

  toggleTheme,

}

}
```

theme хранится в ref, сохраняется в localStorage.

toggleTheme переключает тему.

При загрузке страницы применяется сохранённая или системная тема.

useVacancyFilters.js

```
import { computed, reactive } from 'vue'

import { useJobsStore } from '@/stores/jobs'

import { experienceLabel, formatLabel } from '@/utils/labels'

import { INDUSTRY_LABELS } from '@/constants/industries'

const FILTERS_KEY = 'jobFilters'

const defaults = () => ({ industry: '', salary: 0, experience: '', format: '', keyword:
'' })


export const useVacancyFilters = () => {

  const store = useJobsStore()

  const saved = JSON.parse(localStorage.getItem(FILTERS_KEY) || 'null')

  const filters = reactive({ ...defaults(), ...(saved || {}) })

  const save = () => localStorage.setItem(FILTERS_KEY,
JSON.stringify(filters))

  const reset = () => { Object.assign(filters, defaults()); save() }

  const applyQuickSearch = (keyword, city) => {

    filters.keyword = `${keyword || ''} ${city || ''}`.trim().toLowerCase()

    save()

  }

  const matches = (v) => {

    const salaryOk = !filters.salary || (v.salaryMax || v.salaryMin || 0) >=
filters.salary

    const industryOk = !filters.industry || v.industry === filters.industry
```

```
    const experienceOk = !filters.experience || v.experienceBucket ===
filters.experience

    const formatOk = !filters.format || v.format === filters.format

    const text = `${v.title} ${v.company?.name || ''} ${v.description || ''}
${v.location || ''}`.toLowerCase()

    const keywordOk = !filters.keyword || text.includes(filters.keyword)

    return salaryOk && industryOk && experienceOk && formatOk &&
keywordOk

  }

  return {

    filters,

    filteredVacancies: computed(() =>
store.normalizedVacancies.filter(matches)),

    activeFilters: computed(() => {

      const items = []

      if (filters.industry) items.push(INDUSTRY_LABELS[filters.industry] ||
filters.industry)

      if (filters.salary) items.push(`от ${filters.salary.toLocaleString('ru-RU')} ₽`)

      if (filters.experience) items.push(experienceLabel(filters.experience))

      if (filters.format) items.push(formatLabel(filters.format))

      if (filters.keyword) items.push(`Поиск: ${filters.keyword}`)

      return items

    }),

    applyQuickSearch,

    reset,
```

```
    save,
  }
}
```

Все фильтры вынесены в composable.

filters реактивный.

save/reset управляют localStorage.

filteredVacancies вычисляется на основе state.

Layout

MainLayout.vue

```
<script setup>
import { reactive, computed } from 'vue'
import { Modal } from 'bootstrap'
import ThemeToggle from '@/components/ThemeToggle.vue'
import { useJobsStore } from '@/stores/jobs'
const store = useJobsStore()
const user = computed(() => store.user)
const loginForm = reactive({ email: '', password: '' })
const registerForm = reactive({ fullName: '', email: '', password: '' })
const hideModal = (id) =>
Modal.getInstance(document.getElementById(id))?.hide()


const submitLogin = async () => {
  try {
```

```
    await store.login({ ...loginForm })
    loginForm.email = ''
    loginForm.password = ''
    hideModal('loginModal')
  } catch {
    alert('Ошибка авторизации.')
  }
}
const submitRegister = async () => {
  try {
    await store.register({ ...registerForm })
    registerForm.fullName = ''
    registerForm.email = ''
    registerForm.password = ''
    hideModal('registerModal')
  } catch {
    alert('Ошибка регистрации.')
  }
}
const logout = () => store.logout()
</script>
```

Шапка содержит навигацию по роутам и переключатель темы.

Кнопки «Вход» и «Регистрация» открывают модальные окна Bootstrap.

В методах submitLogin и submitRegister идёт вызов store.

Футер ведёт на вакансии и профиль.

Компоненты

ThemeToggle.vue

```
<template>
  <button
    type="button"
    class="btn btn-outline-light btn-sm"
    :aria-pressed="isDark ? 'true' : 'false'"
    @click="toggleTheme"
  >
    {{ isDark ? 'Светлая' : 'Тёмная' }}
  </button>
</template>
<script>
import { useTheme } from '@/composables/useTheme'
export default {
  name: 'ThemeToggle',
  setup() {
    const { isDark, toggleTheme } = useTheme()
    return { isDark, toggleTheme }
  },
}
```

</script>

Кнопка смены темы использует composable useTheme.

FiltersPanel.vue

<template>

<h5 class="mb-0">Фильтры</h5>

<button class="btn btn-link btn-sm p-0" type="button" @click="$emit('reset')">

Сбросить

</button>

<form @input="$emit('change')" @change="$emit('change')">

<label class="form-label" for="industrySelect">Отрасль</label>

<select id="industrySelect" class="form-select mb-3" v-model="filters.industry">

<option value="">Любая</option>

<option v-for="i in industries" :key="i.value" :value="i.value">

{{ i.label }}

</option>

</select>

<label class="form-label" for="salaryInput">Мин. зарплата, ₽</label>

```
      <input id="salaryInput" class="form-control mb-3" type="number"
v-model.number="filters.salary" />

      <label class="form-label" for="experienceSelect">Опыт</label>

      <select id="experienceSelect" class="form-select mb-3"
v-model="filters.experience">

        <option value="">Любой</option>

        <option value="no">Нет опыта</option>

        <option value="1-3">1–3 года</option>

        <option value="3-6">3–6 лет</option>

        <option value="6+">6+ лет</option>

      </select>

      <label class="form-label" for="formatSelect">Формат</label>

      <select id="formatSelect" class="form-select mb-3"
v-model="filters.format">

        <option value="">Любой</option>

        <option value="office">Офис</option>

        <option value="hybrid">Гибрид</option>

        <option value="remote">Удалённо</option>

      </select>

      <label class="form-label" for="keywordInput">Поиск</label>

      <input

        id="keywordInput"

        class="form-control"

        type="text"
```

```
          placeholder="компания или стек"
          v-model.trim="filters.keyword"
        />
      </form>
    </div>
  </div>
</template>
<script setup>
defineProps({
  filters: { type: Object, required: true },
  industries: { type: Array, required: true },
})
defineEmits(['reset', 'change'])
</script>
```

Форма фильтрации вакансий.

Через v-model изменяет filters.

ActiveFilters.vue

```
<template>
  <div class="d-flex flex-wrap align-items-center gap-2 mb-3">
    <span v-if="!items.length" class="text-muted small">Фильтры не выбраны</span>
    <span v-for="item in items" :key="item" class="pill pill--muted">{{ item }}</span>
  </div>
```

```
</template>

<script>
export default {
  name: 'ActiveFilters',
  props: {
    items: {
      type: Array,
      required: true,
    },
  },
}
</script>
```

Отображает выбранные фильтры в виде плашек.

VacancyCard.vue

```
<template>
  <div class="job-card p-3 h-100 d-flex flex-column">
    <div class="d-flex justify-content-between gap-2 mb-2">
      <h3 class="job-card__title mb-0">{{ vacancy.title }}</h3>
      <span class="badge bg-light text-dark border">{{ vacancy.industryLabel
}}</span>
    </div>
    <p class="text-muted mb-2">
```

```
      {{ vacancy.company?.name }} • {{ vacancy.location }} • {{
formatLabel(vacancy.format) }}

    </p>

    <p class="mb-3">{{ vacancy.description }}</p>

    <div class="mt-auto d-flex flex-column gap-2">

      <div class="d-flex flex-wrap gap-2">

        <span class="badge bg-primary-subtle text-primary fw-semibold">{{
salaryLabel }}</span>

        <span v-if="vacancy.salaryMax" class="badge bg-success-subtle
text-success fw-semibold">

          до {{ vacancy.salaryMax.toLocaleString('ru-RU') }} ₽

        </span>

        <span class="badge bg-secondary-subtle text-secondary">

          {{ experienceLabel(vacancy.experienceBucket) }}

        </span>

      </div>

      <div class="d-flex gap-2">

        <router-link class="btn btn-outline-primary btn-sm"
:to="`/vacancies/${vacancy.id}`">

          Подробнее

        </router-link>

        <button class="btn btn-warning btn-sm" type="button"
@click="$emit('toggle-favorite', vacancy.id)">

          {{ isFavorite ? 'Убрать' : 'В избранное' }}

        </button>
```

```
      </div>
    </div>
  </div>
</template>
<script setup>
import { computed } from 'vue'
import { experienceLabel, formatLabel } from '@/utils/labels'
const props = defineProps({
  vacancy: { type: Object, required: true },
  isFavorite: { type: Boolean, default: false },
})
defineEmits(['toggle-favorite'])
const salaryLabel = computed(() =>
  props.vacancy.salaryMin
    ? `от ${props.vacancy.salaryMin.toLocaleString('ru-RU')} ₽`
    : 'По договоренности',
)
</script>
```

Карточка вакансии с описанием, зарплатой, тэгами.

Кнопка «В избранное» меняет состояние избранного.

ResumeCard.vue

```
<template>
  <div class="card shadow-sm p-3">
    <div class="d-flex align-items-center gap-3">
      <div class="avatar">{{ initials }}</div>
      <div class="min-w-0">
        <p class="text-muted mb-1">Моё резюме</p>
        <h4 class="mb-1">{{ title }}</h4>
        <p class="text-muted mb-0">{{ meta }}</p>
      </div>
    </div>
    <div class="mt-3 d-flex flex-wrap gap-2">
      <span
        v-for="(skill, idx) in skills"
        :key="skill"
        :class="idx === 0 ? 'pill pill--accent' : 'pill pill--muted'"
      >
        {{ skill }}
      </span>
      <span v-if="!skills.length" class="pill pill--muted">Навыки не
указаны</span>
    </div>
    <p class="mt-3 mb-0">{{ aboutText }}</p>
  </div>
```

```
</template>

<script setup>

import { computed } from 'vue'

const props = defineProps({

  profile: { type: Object, default: null },

})

const title = computed(() => props.profile?.position || props.profile?.fullName || 'Моё резюме')

const meta = computed(() => props.profile?.skills || 'Навыки не указаны')

const aboutText = computed(() => props.profile?.about || 'Добавьте описание в профиле')

const initials = computed(() => {

  const name = props.profile?.fullName || 'Моё резюме'

  return name

    .split(' ')

    .filter(Boolean)

    .slice(0, 2)

    .map((p) => p[0].toUpperCase())

    .join('')

})

const skills = computed(() =>

  (props.profile?.skills || '')

    .split(/[,;]/)

    .map((s) => s.trim())
```

```
    .filter(Boolean),
)
</script>
```

Карточка резюме соискателя.

Формирует инициалы, отображает навыки и описание.

Представления (Views)

HomeView.vue

```
<script setup>
import { ref, computed, onMounted } from 'vue'
import { useRouter } from 'vue-router'
import { useJobsStore } from '@/stores/jobs'
import { useVacancyFilters } from '@/composables/useVacancyFilters'
import VacancyCard from '@/components/VacancyCard.vue'
const router = useRouter()
const store = useJobsStore()
const { applyQuickSearch } = useVacancyFilters()
const quickSearch = ref('')
const quickCity = ref('')
const topVacancies = computed(() => store.normalizedVacancies.slice(0, 4))
const favoritesSet = computed(() => store.favoritesSet)
const onQuickSearch = () => {
  applyQuickSearch(quickSearch.value, quickCity.value)
  router.push('/jobs')
```

```
}

const toggleFavorite = (id) => store.toggleFavorite(id)

onMounted(() => store.init())

</script>
```

Детальная карточка вакансии.

Кнопка «Откликнуться» создаёт запись в responses.