САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Бархатова Наталья

Группа

K3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

Задача

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

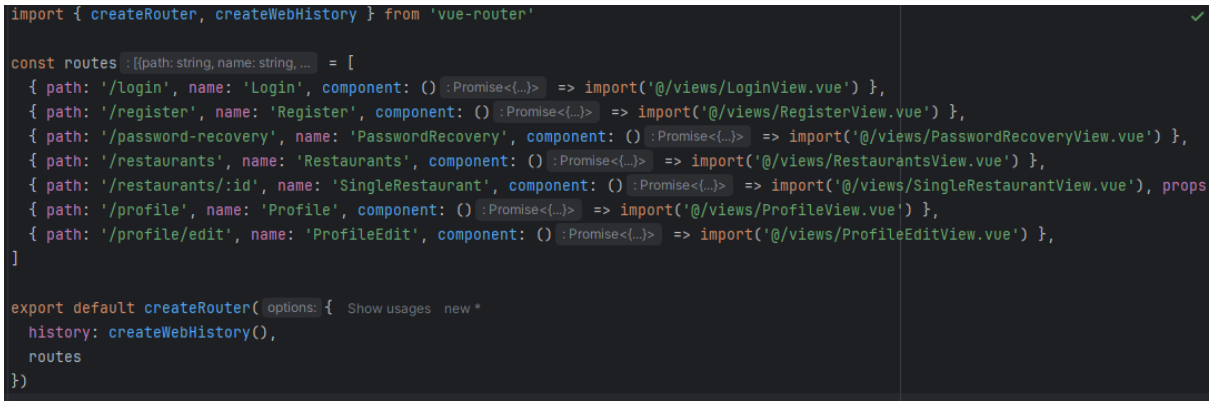
Ход работы

1. **API**

1) Был создан файл instance.js с настроенным экземпляром axios. Это позволило централизовать базовый URL и избежать его дублирования в каждом API классе.
2) Были реализованы четыре API класса для работы с различными сущностями:
   - **auth.js** - управление пользователями
   - **restaurant.js** - работа с ресторанами
   - **user.js** - профиль пользователя
   - **bookings.js** – бронирования
3) В файле api/index.js были созданы экземпляры всех API классов и экспортированы для использования в приложении

2. **Роутер**

В файле router/index.js была настроена маршрутизация приложения с использованием Vue Router.



```
import { createRouter, createWebHistory } from 'vue-router'

const routes : [{path: string, name: string, ... = [
  { path: '/login', name: 'Login', component: () : Promise<{...}> => import('@/views/LoginView.vue') },
  { path: '/register', name: 'Register', component: () : Promise<{...}> => import('@/views/RegisterView.vue') },
  { path: '/password-recovery', name: 'PasswordRecovery', component: () : Promise<{...}> => import('@/views/PasswordRecoveryView.vue') },
  { path: '/restaurants', name: 'Restaurants', component: () : Promise<{...}> => import('@/views/RestaurantsView.vue') },
  { path: '/restaurants/:id', name: 'SingleRestaurant', component: () : Promise<{...}> => import('@/views/SingleRestaurantView.vue'), props
  { path: '/profile', name: 'Profile', component: () : Promise<{...}> => import('@/views/ProfileView.vue') },
  { path: '/profile/edit', name: 'ProfileEdit', component: () : Promise<{...}> => import('@/views/ProfileEditView.vue') },
]

export default createRouter( options: { Show usages  new *
  history: createWebHistory(),
  routes
})
```

3. Pinia

В файле stores/index.js была выполнена инициализация Pinia. В stores/auth.js была реализована логика авторизации, регистрации и восстановления пароля. В stores/profile.js была реализована работа с профилем пользователя и его бронированиями. Геттеры upcomingBookings и pastBookings автоматически фильтруют бронирования по дате.

```
getters: {
  upcomingBookings() {
    const now :Date  = new Date()
    return this.bookings
      .map(b => ({
        ...b,
        restaurantName: this.restaurantsMap[b.restaurantId] || 'Неизвестный ресторан',
        dateTime: new Date( value: `${b.date}T${b.time}`)
      }))
      .filter(b => b.dateTime >= now)
  },

  pastBookings() {
    const now :Date  = new Date()
    return this.bookings
      .map(b => ({
        ...b,
        restaurantName: this.restaurantsMap[b.restaurantId] || 'Неизвестный ресторан',
        dateTime: new Date( value: `${b.date}T${b.time}`)
      }))
      .filter(b => b.dateTime < now)
  }
}
```

В stores/restaurant.js была реализована загрузка списка ресторанов с поддержкой фильтрации.

```
import { defineStore } from 'pinia'
import { restaurantApi } from '@/api'

export const useRestaurantsStore :StoreDefinition<"restaurants", {...}, {...}, {...}>  = defineStore( id: 'restaurants',  options: {  Show usages  new *
  state: () :{...}  => ({
    restaurants: [],
    error: null,
    loading: false
  }),

  actions: {
    async loadRestaurants(filters :{}  = {}) :Promise<void>  {
      try {
        this.loading = true
        const { data } = await restaurantApi.getRestaurants(filters)
        this.restaurants = data
        this.error = null
      } catch (e) {
        console.error(e)
        this.error = 'Не удалось загрузить рестораны'
      } finally {
        this.loading = false
      }
    }
  }
})
```

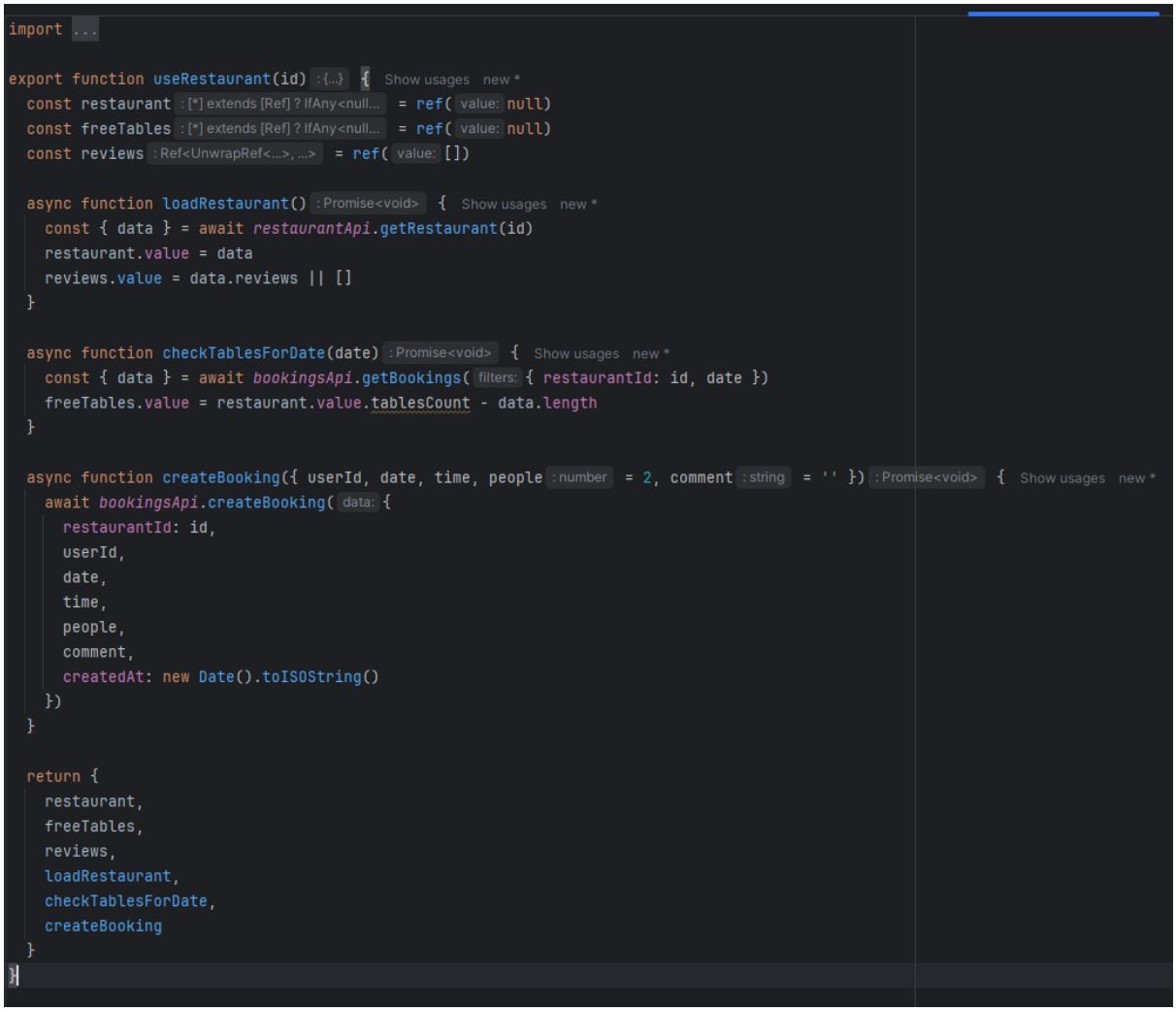
В stores/themeStore.js была реализована система переключения тем:

Тема сохраняется в localStorage и применяется при инициализации приложения через динамическую смену CSS файла в index.html:

```
<link id="theme-style" rel="stylesheet" href="/src/assets/css/theme-light.css">
```

4. Composables

В файле composables/useRestaurant.js была выделена повторяющаяся логика работы с рестораном. Composable инкапсулирует логику загрузки ресторана, проверки доступных столиков и создания бронирования.



```
import ...

export function useRestaurant(id) :{...} {  Show usages  new *
  const restaurant :[*] extends [Ref] ? IfAny<null... = ref( value: null)
  const freeTables :[*] extends [Ref] ? IfAny<null... = ref( value: null)
  const reviews :Ref<UnwrapRef<...>, ...> = ref( value: [])

  async function loadRestaurant() :Promise<void> {  Show usages  new *
    const { data } = await restaurantApi.getRestaurant(id)
    restaurant.value = data
    reviews.value = data.reviews || []
  }

  async function checkTablesForDate(date) :Promise<void> {  Show usages  new *
    const { data } = await bookingsApi.getBookings( filters: { restaurantId: id, date })
    freeTables.value = restaurant.value.tablesCount - data.length
  }

  async function createBooking({ userId, date, time, people :number = 2, comment :string = '' }) :Promise<void> {  Show usages  new *
    await bookingsApi.createBooking( data: {
      restaurantId: id,
      userId,
      date,
      time,
      people,
      comment,
      createdAt: new Date().toISOString()
    })
  }

  return {
    restaurant,
    freeTables,
    reviews,
    loadRestaurant,
    checkTablesForDate,
    createBooking
  }
}
```

5. Компоненты

1) RestaurantGallery.vue - галерея фотографий с модальным просмотром. Компонент использует Bootstrap Modal для отображения увеличенного изображения при клике.
2) ReviewsBlock.vue - блок отзывов посетителей

6. Представления

1) LoginView.vue - страница входа

```vue
<script setup>
import { ref, computed, onMounted } from 'vue'
import { useRouter } from 'vue-router'
import useAuthStore from '@/stores/auth'
import BaseLayout from '@/layouts/BaseLayout.vue'

const router = useRouter()
const authStore = useAuthStore()

const form = ref( value: {
  email: '',
  password: ''
})

const error = computed( getter: () => authStore.error)

const submit = async () => {  Show usages  new *
  const success = await authStore.login(form.value.email, form.value.password)
  if (success) {
    router.push('/restaurants')
  }
}

onMounted( hook: () => {
  authStore.clearError()
})
</script>
```

2) RegisterView.vue - страница регистрации с валидацией совпадения паролей

```vue
<script setup>
const form = ref( value: {
  password: '',
  password_repeat: ''
})

const success = ref( value: false)
const error = computed( getter: () => authStore.error)

const submit = async () => {  Show usages  new *
  if (form.value.password !== form.value.password_repeat) {
    authStore.error = 'Пароли не совпадают'
    return
  }

  success.value = await authStore.register(form.value)

  if (success.value) {
    setTimeout( handler: () => {
      router.push('/login')
    }, timeout: 800)
  }
}

onMounted( hook: () => {
  authStore.clearError()
})
</script>
```

3) PasswordRecoveryView.vue - страница восстановления пароля с поддержкой двух сценариев

```
<script setup>
const submit = async () => {  Show usages  new *
  error.value = null
  success.value = false
  if (!confirmed.value) {
    alert("Подтвердите, что вы владелец аккаунта")
    return
  }
  if (form.value.password !== form.value.password_repeat) {
    error.value = 'Пароли не совпадают!'
    return
  }
  try {
    let result = false

    if (userId && token) {
      result = await auth.updatePassword(userId, token, form.value.password)
    } else {
      if (!form.value.email) {
        error.value = 'Укажите email'
        return
      }
      result = await auth.resetPasswordByEmail(form.value.email, form.value.password)
    }
    if (result) {
      success.value = true
      setTimeout( handler: () => {
        router.push('/login')
      }, timeout: 800)
    } else {
      error.value = auth.error
    }
```

4) RestaurantsView.vue - список ресторанов с фильтрацией

```vue
<script setup>
const filters = ref( value: {
})

const cuisineMap = {
  Italian: 'Итальянская',
  Asian: 'Азиатская',
  French: 'Французская',
  North: 'Северная'
}

const locationMap = {
  Centre: 'Центральный',
  Primorsky: 'Приморский',
  Autozavodsky: 'Автозаводский'
}

const restaurants = computed( getter: () => restaurantsStore.restaurants)

const applyFilters = () => {  Show usages  new *
  const params = {}

  if (filters.value.cuisine) params.cuisine = filters.value.cuisine
  if (filters.value.location) params.location = filters.value.location
  if (filters.value.price) params.price = Number(filters.value.price)

  restaurantsStore.loadRestaurants(params)
}

onMounted( hook: () => {
  restaurantsStore.loadRestaurants()
})
```

5) SingleRestaurantView.vue - детальная страница ресторана:

```
const checkTables = () => {  Show usages  new *
  if (selectedDate.value) checkTablesForDate(selectedDate.value)
}

const submitBooking = async () => {  Show usages  new *
  if (!bookingDate.value || !bookingTime.value) return

  await createBooking( {userId, date, time, people = 2, comment = ""} {
    userId,
    date: bookingDate.value,
    time: bookingTime.value
  })

  checkTablesForDate(bookingDate.value)
  selectedDate.value = bookingDate.value
  showBookingModal.value = false
  bookingDate.value = ''
  bookingTime.value = ''
  showToast.value = true
  setTimeout( handler: () => showToast.value = false,  timeout: 3000)
}

onMounted(loadRestaurant)
</script>
```

6) ProfileView.vue - отображение профиля с бронированиями

```vue
<script setup>

const router = useRouter()
const profileStore = useProfileStore()
const themeStore = useThemeStore()

const defaultAvatar = '/images/icon.png'
const userId = localStorage.getItem( key: 'userId')

const theme = computed( getter: () => themeStore.theme)
const toggleTheme = () => themeStore.toggleTheme()  Show usages  new *

onMounted( hook: async () => {
  themeStore.initTheme()
  if (!userId) {
    router.push('/login')
    return
  }

  try {
    await profileStore.loadProfile(userId)
  } catch (error) {
    console.error('Ошибка загрузки профиля:', error)
    router.push('/login')
  }
})

const user = computed( getter: () => profileStore.user)
const upcomingBookings = computed( getter: () => profileStore.upcomingBookings)
const pastBookings = computed( getter: () => profileStore.pastBookings)
</script>
```

7) **ProfileEditView.vue** - редактирование профиля

```vue
<script setup>
onMounted( hook: async () => {
})

const onAvatarChange = (e) => {  Show usages  new *
  avatarFile = e.target.files[0] || null
  if (avatarFile) {
    const reader = new FileReader()
    reader.onload = () => {
      user.value.avatar = reader.result
    }
    reader.readAsDataURL(avatarFile)
  } else {
    user.value.avatar = currentAvatar
  }
}

const submitProfile = async () => {  Show usages  new *
  try {
    await axios.patch( url: `${API_URL}/users/${userId}`, user.value,  config: {
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`
      }
    })
    router.push('/profile')
  } catch (e) {
    console.error('Ошибка сохранения профиля', e)
    alert('Не удалось сохранить профиль')
  }
}
</script>
```

Вывод

В ходе работы приложение было полностью мигрировано на Vue.js с использованием Vue Router для маршрутизации и Pinia для управления состоянием. Были реализованы централизованные API-классы, composables для повторяющейся логики и компоненты для отображения ресторанов, отзывов и профиля пользователя.