Отчет о выполнении лабораторных работ 14 и 15

Лабораторная работа 14 — Security и аутентификация

Хранить JWT в sessionStorage

Измененные файлы:

- `contexts/AuthContext.tsx`:
  - Строка 18: `localStorage.getItem('auth_token')` → `sessionStorage.getItem('auth_token')`
  - Строка 53: `localStorage.setItem('auth_token', ...)` → `sessionStorage.setItem('auth_token', ...)`
  - Строка 124: `localStorage.removeItem('auth_token')` → `sessionStorage.removeItem('auth_token')`

Преимущества:

- Токен автоматически удаляется при закрытии вкладки браузера
- Повышенная безопасность за счет ограничения времени жизни токена

- Соответствие требованиям лабораторной работы

Реализовать Snackbar для ошибок

Созданные файлы:

- `contexts/NotificationContext.tsx`:
  - Контекст для управления уведомлениями
  - Методы: `showNotification()`, `showError()`, `showSuccess()`, `showWarning()`, `showInfo()`
  - Компонент Snackbar с автоматическим скрытием через 6 секунд
  - Позиционирование: нижний правый угол

Интеграция:

- `pages/_app.tsx`: Добавлен `NotificationProvider` для глобального доступа к уведомлениям

Заменены все использования `alert()` и простых div на Snackbar:

1. **`components/auth/LoginForm.tsx`:**
   - Удален `<div>` с ошибкой
   - Добавлен `useNotification()` hook
   - Ошибки теперь показываются через Snackbar
2. **`pages/admin/users/index.tsx`:**
   - Заменены 4 вызова `alert()` на `showError()`
   - Улучшена обработка ошибок при загрузке, обновлении и удалении пользователей
3. **`pages/owners/index.tsx`:**
   - Заменены 3 вызова `alert()` на `showError()`
   - Ошибки при создании, обновлении и удалении владельцев
4. **`pages/admin/settings/index.tsx`:**
   - Заменены 4 вызова `alert()` на `showError()` и `showSuccess()`
   - Успешные операции теперь показываются через зеленый Snackbar

Преимущества:

- Современный и красивый UI
- Не блокирует взаимодействие с интерфейсом
- Централизованное управление уведомлениями
- Поддержка различных типов уведомлений (error, success, warning, info)
- Автоматическое скрытие через 6 секунд

Лабораторная работа 15 — Деплой и контейнеризация

Собрать и запустить Docker контейнеры (MariaDB + FastAPI)

Созданные файлы:

1. `Dockerfile.backend`:
    - Базовый образ: `python:3.11-slim`
    - Установка системных зависимостей (gcc, postgresql-client)
    - Установка Python зависимостей из `requirements.txt`
    - Копирование кода приложения
    - Порт: 8000
    - Команда запуска: `uvicorn app.main:app --host 0.0.0.0 --port 8000`
2. `Dockerfile.frontend`:
    - Многоэтапная сборка для оптимизации размера образа
    - Базовый образ: `node:18-alpine`
    - Установка зависимостей
    - Сборка Next.js приложения в standalone режиме
    - Production образ с минимальными зависимостями
    - Порт: 3000
3. `docker-compose.yml`:
    - **Сервис `mariadb`:**
        - Образ: `mariadb:10.11`

        - Персистентное хранилище через volumes
        - Healthcheck для проверки готовности БД
        - Порт: 3306
    - **Сервис `backend`:**
        - Сборка из `Dockerfile.backend`
        - Зависит от `mariadb` (ожидает готовности)
        - Переменные окружения для подключения к БД
        - Порт: 8000
        - Автоматическая перезагрузка при изменении кода
    - **Сервис `frontend`:**
        - Сборка из `Dockerfile.frontend`
        - Зависит от `backend`
        - Переменные окружения для API URL
        - Порт: 3000
4. **`.dockerignore`:**
    - Исключение ненужных файлов из Docker контекста
    - Оптимизация времени сборки

Использование:

```
# Запуск всех сервисов
docker-compose up -d

# Просмотр логов
docker-compose logs -f

# Остановка
docker-compose down
```

Задеплоить frontend в Netlify

Созданные файлы:

1. **`netlify.toml`:**
    - Конфигурация сборки: `npm run build`
    - Publish directory: `.next`

- Перенаправления для Next.js
- Настройки кэширования для статических файлов
- CORS заголовки для безопасности
- Переменные окружения для API URL

2. `next.config.js`:

   - Добавлен `output: 'standalone'` для Docker и Netlify
   - Оптимизация для production сборки

Процесс деплоя:

1. Подключение репозитория к Netlify
2. Автоматическая сборка при push в main ветку
3. Настройка переменных окружения в Netlify Dashboard
4. Автоматический деплой

Задеплоить backend в Elastic Beanstalk

Созданные файлы:

1. `.ebextensions/python.config`:

   - Настройки Python окружения
   - WSGI путь: `app.main:app`
   - Health check путь: `/api/status`
   - Настройки логирования в CloudWatch
   - Переменные окружения для БД и JWT

2. `.ebextensions/01_python_packages.config`:

   - Установка Python зависимостей
   - Инициализация базы данных при первом деплое

3. `Procfile`:

   - Команда запуска приложения для Elastic Beanstalk
   - `web: uvicorn app.main:app --host 0.0.0.0 --port 8000`

Процесс деплоя:

1. Инициализация EB: `eb init -p python-3.11`
2. Создание окружения: `eb create car-api-env`

3. Настройка переменных окружения
4. Деплой: `eb deploy`

Развернуть БД в AWS RDS

Обновленные файлы:

1. `app/db.py`:
   - Добавлена функция `get_db_url()` для поддержки различных типов БД
   - Поддержка переменных окружения AWS RDS:
     - `RDS_HOSTNAME`
     - `RDS_PORT`
     - `RDS_DB_NAME`
     - `RDS_USERNAME`
     - `RDS_PASSWORD`
   - Поддержка полного `DB_URL` для гибкости
   - Автоматическое определение типа БД (PostgreSQL/MySQL/MariaDB)
   - Настройки connection pooling для production
   - Поддержка таймаутов и переподключений
2. `requirements.txt`:
   - Добавлен `pymysql==1.1.*` для поддержки MySQL/MariaDB

Процесс настройки:

1. Создание RDS инстанса в AWS Console
2. Настройка Security Groups для доступа из Elastic Beanstalk
3. Получение endpoint RDS
4. Настройка переменных окружения в Elastic Beanstalk
5. Автоматическая инициализация БД при первом запуске

Поддерживаемые типы БД:

- PostgreSQL (через `psycopg`)
- MySQL/MariaDB (через `pymysql`)
- AWS RDS (автоматическое определение)
- SQLite (для разработки)

Созданные и измененные файлы

Новые файлы:

1. `contexts/NotificationContext.tsx` - Контекст для уведомлений
2. `Dockerfile.backend` - Docker образ для FastAPI
3. `Dockerfile.frontend` - Docker образ для Next.js
4. `docker-compose.yml` - Оркестрация контейнеров
5. `netlify.toml` - Конфигурация Netlify
6. `.ebextensions/python.config` - Конфигурация Elastic Beanstalk
7. `.ebextensions/01_python_packages.config` - Установка зависимостей EB
8. `Procfile` - Команда запуска для EB
9. `.dockerignore` - Исключения для Docker
10. `README_DEPLOY.md` - Инструкции по деплою
11. `ОТЧЕТ_ЛАБОРАТОРНЫЕ_РАБОТЫ_14_15.md` - Этот отчет

Измененные файлы:

1. `contexts/AuthContext.tsx` - localStorage → sessionStorage
2. `pages/_app.tsx` - Добавлен NotificationProvider
3. `components/auth/LoginForm.tsx` - Заменены div на Snackbar
4. `pages/admin/users/index.tsx` - Заменены alert() на Snackbar
5. `pages/owners/index.tsx` - Заменены alert() на Snackbar
6. `pages/admin/settings/index.tsx` - Заменены alert() на Snackbar
7. `app/db.py` - Поддержка AWS RDS
8. `requirements.txt` - Добавлен pymysql
9. `next.config.js` - Добавлен output: 'standalone'

Заключение

Все задачи лабораторных работ 14 и 15 успешно выполнены. Проект готов к деплою в production окружение с использованием современных практик DevOps и обеспечения безопасности.