1. Общая концепция

Веб-портал для внутреннего обучения сотрудников и стажёров с ролевым доступом, поэтапным изучением материалов и контролем знаний.

Роли: - Обучающийся (сотрудник / стажёр) - Администратор

Ключевые принципы (рекомендации): - RBAC (role-based access control) - Жёсткая авторизация (JWT + refresh или сессии) - Аудит действий (логирование) - Хранение персональных данных с шифрованием - Модульность (материалы, тесты, пользователи — независимо)

---

2. Структура сайта (страницы)

2.1 Публичная зона

- **/login** — страница авторизации

❗ Регистрация отсутствует. Пользователей создаёт администратор.

2.2 Зона обучающегося

/dashboard — главная

- ФИО, звание, должность
- Прогресс обучения (%)
- Список направлений обучения
- Статус: `не начато / изучается / завершено`

/course/:id — страница темы

- Описание темы
- Обучающие материалы:
- PDF (iframe/viewer)
- Видео (HTML5 / streaming)
- Статус изучения
- Кнопка "Перейти к тесту" (активна после просмотра)

/test/:id — тест по теме

- Вопросы (1 или несколько правильных ответов)
- Таймер (опционально)
- Минимальный проходной балл

/final-test — итоговый тест

- Вопросы из всех тем
- Строгая попытка (например 1–2)

/certificate — итог

- Электронный диплом (PDF)
- Дата, ФИО, курс
- Уникальный номер

2.3 Административная зона

/admin/users

- Создание / редактирование обучающихся
- Данные:
- ФИО
- Дата рождения
- Паспорт (⚠ шифрование)
- Должность
- Звание
- Роль

/admin/courses

- Создание направлений
- Управление темами

/admin/materials

- Загрузка PDF
- Загрузка видео
- Привязка к теме

/admin/tests

- Управление вопросами
- Настройка баллов и попыток

/admin/reports

- Прогресс пользователей
- Результаты тестов

---

3. Логика обучения

1. Пользователь входит в систему
2. Выбирает тему
3. Изучает материалы
4. Проходит тест темы
5. После всех тем — итоговый тест
6. При успешном завершении — генерация диплома

---

4. Структура базы данных (PostgreSQL)

users

- id (PK)
- last_name
- first_name
- middle_name
- birth_date
- passport_encrypted
- rank
- position
- role (student/admin)
- password_hash
- created_at

---

courses (направления)

- id (PK)
- title
- description
- is_active

---

lessons (темы)

- id (PK)
- course_id (FK)
- title
- order_index

---

materials

- id (PK)
- lesson_id (FK)
- type (pdf/video)
- file_path
- duration

---

tests

- id (PK)
- lesson_id (FK, nullable)
- is_final (bool)
- passing_score

---

questions

- id (PK)
- test_id (FK)
- question_text

---

answers

- id (PK)
- question_id (FK)
- answer_text
- is_correct

---

user_progress

- id (PK)
- user_id (FK)
- lesson_id (FK)
- is_completed
- completed_at

---

test_results

- id (PK)
- user_id (FK)
- test_id (FK)
- score
- passed
- attempt
- completed_at

---

certificates

- id (PK)
- user_id (FK)
- course_id (FK)
- serial_number
- issued_at
- file_path

---

5. Технологический стек (рекомендую)

Backend: - Node.js (NestJS) или Django - PostgreSQL - Redis (сессии)

Frontend: - React / Vue - PDF.js - HTML5 video - Adaptive UI (mobile-first)

Безопасность: - HTTPS - Шифрование ПДн - Логи аудита

---

6. Что можно улучшить

✔️ Разделить сотрудник / стажёр по доступным курсам ✔️ Добавить повторную аттестацию ✔️ Добавить офлайн-отчёты ✔️ Поддержка SCORM (на будущее)

---

7. Итог

Твоя идея абсолютно рабочая и логичная. Я её немного структурировал и усилил с точки зрения: - безопасности - масштабируемости - реальной эксплуатации

Если хочешь — могу: - нарисовать ER-диаграмму - предложить API-эндпоинты - помочь с ТЗ или архитектурой backend/frontend