Техническая документация к практике по теме «Генерация документов для защит ВКР в Practice Grading »

PR: https://github.com/yurii-litvinov/PracticeGrading/pull/11

Документ описывает внесённые изменения **по слоям**: Backend (API), Frontend, База данных, Тесты.

---

1) Backend (ASP.NET Core, C#)

Backend реализован на платформе **ASP.NET Core** (проект `PracticeGrading.API`). Ключевые изменения: управление членами комиссии (CRUD + поиск) и расширенная генерация пакета документов.

1.1. Эндпоинты управления членами комиссии

Файл: `PracticeGrading.API/Endpoints/MembersEndpoints.cs`.

Все эндпоинты требуют авторизации с админ ролью.

GET `/members`

Назначение: поиск членов комиссии по имени (постранично).

Параметры query:

- `searchName` (string) — строка поиска;
- `offset` (int, default 0) — смещение;
- `limit` (int, default 0) — сколько максимум пользователей возвращать.

Ответ `200 OK`:

```
{
  "members": [ /* массив MemberDto */ ],
  "hasMore": true|false
}
```

POST `/members`

Назначение: создать нового члена комиссии.

Тело запроса: `MemberRequest` (см. `PracticeGrading.API/Models/Requests/MemberRequest.cs`).

Ответ: 200 OK

Возвращается **JSON** с идентификатором **добавленного пользователя** в поле `Id`.

Пример:

```
{ "Id": 123 }
```

PUT `/members`

Назначение: обновить данные члена комиссии.

Тело запроса: `MemberRequest`.

Ответ: `200 OK`.

DELETE `/members`

Назначение: удалить члена комиссии по идентификатору.

Параметры query:

- `id` (int) — идентификатор члена комиссии.

Ответ: `200 OK`.

1.2. Изменения генерации документов для встреч/заседаний

Файл: `PracticeGrading.API/Endpoints/MeetingEndpoints.cs`.

Уточнение выдачи данных встречи в зависимости от роли

Метод `GetMeeting(...)` возвращает полную информацию о членах комиссии для пользователя с ролью администратора, а для членов комиссии — только их ФИО.

Генерация архива документов (ZIP)

В `GetDocuments(...)` расширены входные параметры: вместо передачи председателя как строки используется `chairmanId` и дополнительная информация (`chairmanOrder`, `secretary`).

Результат: формируется ZIP-архив с документами, имя архива берётся из `meeting.Info` (или используется значение по умолчанию, если поле пустое).

1.3. Модуль генерации документов (`DocumentsGenerator`)

Генерация документов реализована на backend в виде отдельного класса `DocumentsGenerator` (`PracticeGrading.API/Integrations/DocumentsGenerator.cs`). Модуль формирует комплект DOCX-документов по заранее подготовленным шаблонам, подставляя данные встречи, членов комиссии и студенческих работ. Для работы с DOCX используется библиотека **NPOI** (`XWPFDocument`).

Входные данные и инициализация

Экземпляр `DocumentsGenerator` создаётся на основе:

- `meeting: MeetingDto` — данные заседания (дата/время, информация о заседании, список членов комиссии `Members`, список работ `StudentWorks`);
- `chairman: MemberDto` — председатель (отдельно передаётся для документов, где требуется выделение председателя);
- `chairmanOrder: string` — приказ о назначении председателя;
- `secretary: string` — ФИО секретаря.

При создании экземпляра модуль:

- извлекает номер комиссии и параметры образовательной программы из `meeting.Info` (по префиксу `ГЭК` и фиксированной длине номера), используя `MajorInfoDictionary`;
- формирует строковое представление даты (`ru-RU`, формат `d MMMM yyyy г.`) и времени (`HH:mm`);
- вычисляет значения `major`, `educationalProgram`, `academicDegree` и другие поля, используемые во всех шаблонах.

Общий принцип заполнения шаблонов

Каждый метод генерации принимает **Stream шаблона**, открывает его как `XWPFDocument`, заменяет плейсхолдеры вида `[placeholder]`, а затем (при необходимости) дополняет таблицы/параграфы динамически создаваемыми строками.

Замену плейсхолдеров выполняют методы:

- `ReplacePlaceholdersInParagraphs(...)` — замена в параграфах документа;
- `ReplacePlaceholdersInTables(...)` — замена в таблицах;

Для форматирования текста используются единые настройки:

- шрифт `Times New Roman`, размер `12`.

Генерируемые документы и методы

1. **Ведомость** — файл: `Ведомость ВКР ГЭК {commissionNumber}.docx`
2. **Оценочный лист** — файл: `{member.Name} оценочный лист ГЭК {commissionNumber}.docx`
3. **Согласие на обработку персональных данных** — файл: `{member.Name} Согласие на обработку персональных данных.docx`
4. **Отчёт председателя** — файл: `Отчёт председателя ГЭК {commissionNumber}.docx`
5. **Протокол защиты** — файл: `Протокол защиты {work.StudentName}.docx`
6. **Итоговый протокол** — файл: `Протокол итоговый {commissionNumber}.docx`

Вспомогательная логика

- `ProcessMeetingInfo()`:
    - извлекает номер комиссии из `meeting.Info` по префиксу `ГЭК` и длине `NumberLength`;

    - по коду направления находит сведения в `MajorInfoDictionary`;
    - извлекает год набора из `meeting.Info` и подставляет в строку образовательной программы.

- `GetSurnameWithInitials(fullName)`:
    - переводит “Фамилия Имя Отчество” в формат “Фамилия И.О.” (если частей 2 — одна инициала).

- `GenerateMemberInfoText(...)`:
    - формирует текстовые блоки по члену комиссии (ФИО, информация RU, email, телефон), для председателя добавляет строку про приказ.

- Табличная генерация:
    - `CreateStatementTableHeader(...)` и `FillStatementStudentTable(...)` — динамика таблицы ведомости.
    - `FillGradingSheetStudentTable(...)` — динамика таблицы оценочного листа.

Замечания по данным

Модуль предполагает, что:

- `meeting.Info` содержит строку с `ГЭК` и номером комиссии в ожидаемом формате, иначе бросается `ArgumentException`;
- в `meeting.Members` присутствуют члены комиссии, а председатель передан отдельно (используется для выделения в документах).

Пакетная генерация

Верхнеуровневая сборка комплекта документов для встречи выполняется вне `DocumentsGenerator` (в endpoint’ах), где **все документы генерируются параллельно**, после чего сформированные файлы добавляются в ZIP-архив.

2) Frontend (React + TypeScript)

Frontend реализован на **React + TypeScript** (папка `frontend/`). Задача изменений: добавить на UI функциональность для управления членами комиссии и привязки членов комиссии к встречам, а также собрать недостающие данные для генерации документов.

Новые/изменённые файлы, связанные с управлением членами комиссии:

- страница: `frontend/src/pages/Members.tsx`
- хук: `frontend/src/hooks/useMemberSearch.ts`
- компоненты:
    - `frontend/src/components/MemberList.tsx`
    - `frontend/src/components/MemberModal.tsx`
    - `frontend/src/components/MemberSearchDropdown.tsx`
    - `frontend/src/components/MemberSearchList.tsx`
- API-клиент: `frontend/src/services/ApiService.ts`

- интеграция со страницами встреч: `MeetingFormPage.tsx`, `ViewMeetingPage.tsx`.

2.1 Страница Members

Страница управления членами комиссии: поиск, открытие карточки, добавление/редактирование/удаление через модальное окно.

Файл: `frontend/src/pages/Members.tsx`.

Состояния:

- `search: string` — строка поиска (передаётся в `MemberSearchList` и используется как дефолтное имя в модалке при добавлении).
- `selectedMember: Member | null` — выбранный элемент списка (для редактирования).
- `isModalOpen: boolean` — открыто ли модальное окно.
- `isModalRequestLoading: boolean` — флаг загрузки для операций add/update/delete.
- `reloadKey: number` — “триггер” перезагрузки списка (инкремент после успешных CRUD-операций).

Логика:

- клик по члену комиссии → `selectedMember` + открытие `MemberModal`;
- “Добавить” → открытие `MemberModal` с `member={ name: search }`;
- `onSave` в модалке:
  - если `member.id` есть → `updateMember(member)`,
  - иначе → `addMember(member)`,
  - затем `reloadKey++` и закрытие модалки;
- `onDelete` → `deleteMember(id)`, затем `reloadKey++` и закрытие.

2.2 Хук useMemberSearch

Хук для поиска членов комиссии с пагинацией и дебаунсом. Делает запросы через `searchMembers(...)`.

Файл: `frontend/src/hooks/useMemberSearch.tsx`.

Сигнатура:

- `useMemberSearch(searchName: string)`

Поведение:

- нормализует поисковую строку через `trim()`;
- делает запросы с `pageSize = 10`;
- использует `AbortController`: предыдущий запрос отменяется при новом поиске;
- дебаунс ввода: запрос выполняется через `300ms` после изменения `searchName`;
- защита от лишних запросов: если `trim(searchName)` не изменился — повторный запрос не выполняется.

Возвращает:

- `members: Member[]` — текущие результаты;
- `isLoading: boolean` — идёт ли загрузка;
- `hasMore: boolean` — есть ли следующая страница;
- `offset: number` — текущий сдвиг (сколько уже загружено);
- `loadMore: () => void` — догрузить следующих `limit` членов комиссии (если `hasMore` и не `isLoading`);
- `reload: () => void` — перезагрузить с `offset = 0` (используется после CRUD).

2.3. Компонент `MemberList`

Файл: `frontend/src/components/MemberList.tsx`.

Компонент для отображения списка членов комиссии в виде карточек. Поддерживает состояния загрузки, ограничение высоты со скроллом, догрузку элементов, а также опциональные действия над карточками (клик/удаление).

Props (`MemberListProps`):

- `members: Member[]` — список отображаемых членов комиссии.
- `isLoading?: boolean` (default `false`) — признак загрузки.
- `hasMore?: boolean` (default `false`) — есть ли ещё члены комиссии.
- `onLoadMore?: () => void` — обработчик кнопки «Показать ещё».
- `clickable?: boolean` (default `false`) — делает карточки кликабельными.
- `onMemberClick?: (member: Member) => void` — обработчик клика по карточке.
- `removable?: boolean` (default `false`) — отображает кнопку удаления на карточках.
- `onMemberRemove?: (id: number | undefined) => void` — обработчик удаления.
- `maxHeight?: string | number | 'unlimited'` (default `'300px'`) — ограничение высоты списка (включает скролл).
- `showLoadMore?: boolean` (default `true`) — показывать ли кнопку «Показать ещё».

2.4. Компонент `MemberModal`

Файл: `frontend/src/components/MemberModal.tsx`.

Модальное окно для просмотра/добавления/редактирования/удаления члена комиссии. Поддерживает read-only, валидацию полей и состояние загрузки.

Props (`MemberModalProps`):

- `member: Member` — данные члена комиссии (для редактирования/просмотра) или объект для создания.
- `isOpen: boolean` — открыто ли модальное окно.

- `onClose: () => void` — закрыть модальное окно.
- `onSave?: (member: Member) => void | Promise<void>` — сохранить изменения / добавить нового члена комиссии.
- `onDelete?: (memberId: number) => void | Promise<void>` — удалить члена комиссии.
- `isLoading?: boolean` — признак выполнения операции (блокирует элементы управления/кнопки).
- `readOnly?: boolean` — режим «только просмотр».

2.5. Компонент `MemberSearchDropdown`

Файл: `frontend/src/components/MemberSearchDropdown.tsx`.

Поле поиска с выпадающим списком результатов для выбора члена комиссии. Использует `useMemberSearch(value)`, поддерживает догрузку и закрытие по клику вне компонента.

Props (`MemberSearchDropdownProps`):

- `value: string` — строка поиска.
- `onChange: (value: string) => void` — вызывается при вводе текста.
- `onSelect: (member: Member) => void` — вызывается при выборе члена комиссии из списка.
- `placeholder?: string` — плейсхолдер (default `"Поиск членов комиссии..."`).

2.6. Компонент `MemberSearchList`

Файл: `frontend/src/components/MemberSearchList.tsx`.

Компонент «поиск + список результатов» для страницы управления членами комиссии. Использует `useMemberSearch(value)`, поддерживает кнопку «Добавить» и перезагрузку результатов.

Props (`MemberSearchListProps`):

- `value: string` — текущая строка поиска.
- `onChange: (value: string) => void` — обработчик изменения строки поиска.
- `clickable?: boolean` (default `true`) — делает элементы списка кликабельными.
- `onMemberClick?: (member: Member) => void` — обработчик клика по элементу списка.
- `onAddClick?: () => void` — обработчик кнопки «Добавить».
- `reloadKey?: number` — при изменении вызывает перезагрузку результатов.
- `disabled?: boolean` (default `false`) — отключает инпут и кнопку «Добавить».
- `placeholder?: string` — плейсхолдер инпута (default `"Поиск членов комиссии..."`).
- `maxHeight?: string | number | 'unlimited'` — ограничение высоты списка (передаётся в `MemberList`).

2.7 API-клиент (`frontend/src/services/ApiService.ts`)

В `ApiService.ts` добавлены методы для работы с членами комиссии через backend API:

- `searchMembers(searchName: string, offset: number, limit: number, signal?: AbortSignal)` — поиск членов комиссии с пагинацией (используется в `useMemberSearch`, поддерживает отмену запроса через `AbortSignal`).
- `updateMember(member: Member)` — обновление данных члена комиссии (`PUT /members`).
- `addMember(member: Member)` — добавление нового члена комиссии (`POST /members`).
- `deleteMember(id: number)` — удаление члена комиссии (`DELETE /members?id=...`).

Также **обновлён** метод `getDocuments(...)`: добавлены параметры `secretary` и `chairmanOrder`, а председатель теперь передаётся как `chairmanId` (идентификатор пользователя) вместо строки.

2.8 Интеграция со страницами встреч

`MeetingFormPage.tsx`

На странице редактирования/создания встречи прежний список членов комиссии заменён на переиспользуемый компонент `MemberList`

`ViewMeetingPage.tsx`

На странице просмотра встречи:

- добавлен поиск членов комиссии с выпадающим списком через `MemberSearchDropdown` (для выбора/добавления в состав встречи);
- для отображения текущего состава используется `MemberList`;
- в модальном окне, открывающемся по кнопке «Сгенерировать документы», добавлены поля **«Секретарь»** и **«Приказ»** (для передачи данных в соответствующий эндпоинт на бэкэнде).

3) База данных (схема и отношения)

Изменения внесены в слой данных (`PracticeGrading.Data`), включая `init_database.sql`, сущности и репозитории.

Суть изменения:

- связь **встреча ↔ члены комиссии** переведена на **many-to-many**, чтобы один и тот же член комиссии мог участвовать в нескольких встречах;
- расширены данные пользователя/члена комиссии так, чтобы они покрывали требования генерации документов; в частности добавлены поля с информацией о члене комиссии **на русском и на английском языках**, а также **email** и **телефон**.

4) Tests

В PR обновлены существующие тесты под новую модель данных и добавлены новые тесты для новых сценариев. По результатам разработки — все тесты (и обновлённые, и добавленные) успешно проходят.

4.1. Backend тесты

- Endpoint-тесты:
  - `PracticeGrading.Tests/EndpointsTests/MembersEndpointsTests.cs` — сценарии поиска/добавления/редактирования/удаления членов комиссии через API (в т.ч. проверяется взаимодействие API ↔ БД);
  - `PracticeGrading.Tests/EndpointsTests/MeetingEndpointsTests.cs` — адаптация под изменения выдачи встречи/генерации документов.
- Интеграционные тесты генерации документов:
  - `PracticeGrading.Tests/IntegrationsTests/DocumentsGeneratorTests.cs` — генерация DOCX выполняется и сравнивается с эталонными файлами из `PracticeGrading.Tests/TestData`.

Примеры эталонных файлов в `TestData`:

- `expected_statement.docx`, `expected_report.docx`, `expected_protocol.docx`;
- примеры «согласий», «оценочных листов» и «протоколов защиты» (DOCX).

4.2. Frontend e2e (Playwright)

Файл: `frontend/tests/tests.spec.ts`.

- **обновлён** сценарий `create meeting`: тест изменён с учётом нового механизма выбора члена комиссии (поиск и выбор из выпадающего списка на форме встречи);
- **добавлены** e2e-сценарии для управления членами комиссии через интерфейс:
  - создание и удаление члена комиссии (`create and delete new user`);
  - редактирование данных члена комиссии (`edit user`).