# Сценарий презентации программы **Agregator** (30 минут)

## Вступление (0–5 мин)

**Что обсуждаем:** Описываем типичную проблему, с которой сталкиваются преподаватели, студенты и исследователи в работе с материалами.

**Основная идея:** У большинства из нас со временем накапливаются сотни файлов – статьи, книги в PDF, презентации, лекции, изображения, аудиозаписи. Они хранятся в разных папках на компьютере или в облаке, без единой системы. Найти нужный документ бывает сложно: нужно помнить, как назывался файл, в какой папке лежит, или перерывать множество папок вручную. Метаданные (авторы, год, тема) часто не заполнены или неполны, а значит, поиск по ним не работает. В итоге **научные и учебные материалы хранятся хаотично, не систематизированы, и их трудно искать и использовать повторно**.

Расскажите историю, близкую аудитории: например, студент не может быстро найти конспект прошлой лекции к экзамену, или преподаватель тратит часы, собирая разбросанные по разным местам файлы для курса. Подведите к мысли, что нужен единый инструмент для хранения и удобного поиска по всем этим данным.

* **A)** *Упрощённо:* Сфокусируйтесь на чувстве **фрустрации**: «Мы все были в ситуации, когда срочно нужен определённый материал, а он потерян среди бесчисленных папок». Опишите проблему живо и просто, без технических деталей – говорите о **перегруженном рабочем столе**, о **долгих поисках по компьютеру**, о **хаосе в файлах**, с которым знаком каждый. Это поможет студентам и гуманитариям узнать себя в описании проблемы и заинтересоваться решением.
* **B)** *Технически:* Можно добавить пару конкретных деталей: объём типичной библиотеки (например, «у исследователя N гигабайт PDF-файлов»), отсутствие единой системы индексации, сложности с существующими инструментами (скажем, ограниченный поиск в Windows, либо необходимость вручную наводить порядок). Упомяните, что **файлы бывают разных форматов** (PDF, DOCX, изображения, аудио) и без специальной обработки по ним невозможно полноценно искать информацию. Это подготовит IT-специалистов к восприятию технического решения, подчеркнув **масштаб проблемы**.

Завершите вступление формулировкой задачи: *«Нужен инструмент, который объединил бы все эти разрозненные материалы, автоматически привёл их в порядок и позволил мгновенно находить нужную информацию. Представляем Agregator – решение именно для этой проблемы.»*

## Назначение **Agregator** (5–8 мин)

**Что обсуждаем:** Зачем создан Agregator, какие задачи он решает, и **почему выбрана локальная обработка файлов вместо облачных решений**.

Начните с краткого определения: **Agregator — это локальное веб‑приложение для хранения, обогащения и интеллектуального поиска по научным материалам** (статьям, диссертациям, презентациям, изображениям, аудио и др.)[[1]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L1-L5). Подчеркните, что Agregator работает *офлайн*, на компьютере пользователя или в локальной сети, и предназначен для индивидуальных исследователей или команд, которые хотят работать со своей библиотекой материалов без передачи данных внешним сервисам[[2]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L3-L6).

Объясните **почему локально**: - **Конфиденциальность и контроль данных:** Все данные остаются у пользователя. Это важно для исследователей и преподавателей, которые работают с неопубликованными материалами, внутренними документами или просто ценят приватность. В Agregator **все вычисления выполняются на локальной машине**, и доступ регулируется ролями и списками доступа[[2]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L3-L6). Таким образом, вы не рискуете, что ваши документы утекут в интернет. - **Автономность:** Поскольку система не зависит от интернета, ей можно пользоваться в поездке, на конференции, в аудитории с плохим Wi-Fi или вообще на закрытых контурах (например, в учебных классах, где нет доступа к внешним сервисам). **Agregator не требует облака**, и это плюс для тех, кто хочет стабильной работы независимо от внешних факторов. - **Неограниченность и экономия:** Отсутствуют ограничения, связанные с тарифами облачных хранилищ или API. Вы не платите за каждый запрос к ИИ или за дополнительное место – программа развернута у вас, и вы можете хранить столько, сколько позволяет ваш диск, и искать столько, сколько нужно. - **Производительность на месте:** Для больших коллекций файлов локальная индексация и поиск могут быть быстрее и настраиваемее под ваши задачи. Вы сами контролируете параметры обработки данных (например, какие модели ИИ использовать для анализа) – это гибче, чем готовые облачные решения.

Приведите контраст с облачными решениями: например, «Почему бы не использовать Google Диск или Mendeley?» – ответ: у облака есть ограничения (интернет, приватность, нет специальных AI-функций для поиска внутри ваших документов и генерации ответов). Agregator же **специально заточен под научные и учебные материалы**: он **автоматически извлекает из файлов текст и метаданные, классифицирует их, связывает между собой и позволяет делать интеллектуальный поиск с учетом содержания** – возможностей, которых обычно нет «из коробки» в облачных дисках.

После мотивации расскажите коротко, *как Agregator решает проблему из вступления:* он собирает все ваши материалы в одном месте, наводит в них порядок и дает умные инструменты для работы с ними. Это подведет к следующей части – демонстрации возможностей.

* **A)** *Упрощённо:* Сфокусируйтесь на выгодах для конечного пользователя. Например: «Agregator – это как ваша личная библиотека и поисковая система, которая всегда под рукой. Вам не нужно быть айтишником, чтобы им пользоваться – просто загружаете файлы, а дальше система сама всё упорядочит и позволит искать, как в Google, только по вашим материалам». Избегайте технического жаргона, подчеркните **простоту использования** и практическую пользу.
* **B)** *Технически:* Здесь можно упомянуть архитектуру в общих чертах, чтобы зацепить интерес IT-аудитории. Например: «Agregator – это локальный веб-сервис на Flask/React с SQLite базой, который сканирует директории, индексирует файлы, использует встроенные модели (OCR, NLP) и интегрируется с LLM для расширенного поиска». Отметьте, что **система модульная и расширяемая**, open-source (исходный код открыт на GitHub), и что **данные хранятся структурировано в SQL базе** (ORM SQLAlchemy)[[3]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L42-L50)[[4]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L61-L68). Это покажет техническим специалистам, что под капотом серьёзное решение, сравнимое по возможностям с корпоративными системами, но доступное каждому.

Перейдите к сути: *«Давайте посмотрим на основные функции Agregator в действии.»*

## Демонстрация основных функций Agregator (~8–23 мин)

**Что делаем:** Пошагово показываем в интерфейсе программы ключевые функции, перечисляя и поясняя каждую. Ниже – план демонстрации с пояснениями **что говорить** и **что показывать на экране**.

Перед началом убедитесь, что у вас запущено приложение Agregator и загружено несколько примерных материалов (несколько научных статей в PDF, пара презентаций, изображений или аудио-файлов) чтобы было на чем показать возможности.

**1. Единое хранилище и организация коллекций (примерно 2 мин):**

* **Говорим:** В Agregator все файлы хранятся **в едином каталоге**, доступном через веб-интерфейс. Слева – панель *«Коллекции»*, где можно создавать тематические подборки или разделять материалы по проектам/курсам. Это похоже на папки, но с дополнительными возможностями управления доступом. Расскажите, что Agregator **индексирует указанную папку на диске** (SCAN\_ROOT) и автоматически подтягивает все файлы в базу данных[[5]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L33-L33). Все метаданные файлов приводятся к единому виду (имена авторов, названия, год издания – вычищаются дубликаты, устанавливается единый формат), что создаёт порядок. В результате пользователь получает **единый упорядоченный каталог всех своих материалов**, которыми можно управлять через браузер.
* **Показываем:** Начните с главной страницы “Каталог” – на экране видны загруженные документы. Продемонстрируйте структуру: **список/плитка материалов** по центру, и панель коллекций слева. Возможно, к этому моменту вы уже импортировали файлы – покажите, как они сгруппированы. Например: *«Вот наши коллекции: ‘Курсовая 2025’, ‘Лекции по истории’, ‘Исследование по ИИ’. Мы можем переключаться между ними, чтобы увидеть соответствующие материалы.»* Кликните на коллекцию – список файлов фильтруется по ней. Обратите внимание на карточку материала: обычно там отображается название, автор, год, тип файла, размер и, возможно, краткая аннотация или отрывок текста (если есть). Акцентируйте, что **больше не нужно помнить, в какой папке файл – всё доступно из одного окна**.
* **A)** *Упрощённо:* Не вдаваясь в детали, скажите, что **создать коллекцию очень просто** – например, для каждого проекта или учебного курса – и **переместить** файлы в нужную коллекцию можно прямо тут (drag-and-drop или через интерфейс). Это понятные аналогии (папки/метки), что поможет гуманитариям: *«Как в библиотеке: у каждой книги своё место на полке.»* Избегайте технических терминов, скажите просто: *«Agregator берёт на себя работу библиотекаря – аккуратно раскладывает ваши материалы по полочкам.»*
* **B)** *Технически:* Упомяните, что **под капотом** при индексации создаётся запись в базе данных для каждого файла. Можно добавить: *«Agregator сохраняет материалы в SQLite базу вместе с метаданными, а файлы остаются на диске. Каждая коллекция имеет свой ID и ACL – списки доступа (viewer/editor/owner), так что можно разграничивать права пользователей*[*[6]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L36-L37)[*[7]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L2-L2)*.»* То есть, техническому слушателю отметьте, что это не просто плоское хранилище – **реализована полноценная модель доступа и хранения**, близкая к профессиональным системам документооборота.

**2. Автоматическая обработка файлов (примерно 2–3 мин):**

* **Говорим:** Теперь продемонстрируем, *как добавляются новые материалы*. Подчеркните, что Agregator умеет **сам анализировать содержимое файлов**. Скажем, у нас есть PDF-статья и изображение страницы книги – добавим их. После добавления **программа автоматически:**
* **Извлекает текст** из документов. Для PDF и DOCX – просто читает содержимое, для изображений или отсканированных PDF – делает OCR (распознавание текста), для аудио/видео – **расшифровывает речь** (через встроенную модель Faster Whisper).
* **Определяет тип материала** – статья, презентация, изображение, диссертация и т.д. (с помощью правил и, при необходимости, LLM-подсказки).
* **Выделяет ключевые слова и генерирует теги** – чтобы файл потом легче было найти. Используются как простые алгоритмы (часто встречающиеся слова), так и модель LLM, чтобы извлечь смысловые ключевые слова.
* **Сохраняет все эти данные** в базу и обновляет индекс поиска.

То есть, **никакой ручной работы – Agregator сам “обогащает” ваши материалы метаданными и создает для них миниатюры/превью**[[8]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L33-L36)[[9]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L84-L88).

* **Показываем:** Проделайте **импорт файла**. Нажмите кнопку «Импорт» (или перетащите файл в окно приложения). На экране, вероятно, появится прогресс или запись в журнале импорта. Рассказывайте: *«Мы добавляем новый файл – Agregator сразу извлекает из него текст и информацию...»*. Если интерфейс позволяет, откройте **лог операций** или страницу «Импорт», где видно, что задача по сканированию файла поставлена в очередь и выполнена[[10]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L74-L79). Через несколько секунд обновите каталог – **новый документ появился**. Покажите его карточку: там уже подтянулось название (если было в PDF), автор, год – даже если в исходном файле это не было заполнено идеально. Пролистайте свойства этого файла (может быть, открыв карточку или окно информации): вы увидите автоматически сгенерированные **теги** и, возможно, **аннотацию или отрывок текста** (первые строки)[[11]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L83-L88). Если файл был картинкой/PDF без текста – укажите, что текст распознан OCR. Если аудио – что доступна транскрипция. **Важно:** акцентируйте, что **вручную ничего этого не вводили** – система сама проанализировала материал.
* **A)** *Упрощённо:* Объясните простыми словами: *«Agregator сам читает документы вместо вас.»* Приведите аналогию: *«Представьте, у вас есть помощник, который берёт новый учебник, сам выписывает из него важные термины, автора, год, и кладёт его на правильную полку.»* Подчеркните выгоду: *«Вам больше не надо помнить, что и где лежит –* *достаточно загрузить файл, а дальше Agregator сделает всю тяжёлую работу.»* Избегайте внутренних терминов (OCR, LLM) – достаточно сказать «распознаёт текст с картинок», «понимает аудио».
* **B)** *Технически:* Здесь можно немного пройтись по технологиям: *«Agregator использует Tesseract для OCR, а для аудио – модель* *Faster Whisper* *для транскрибации речи*[*[12]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L34-L35)*. Он также умеет звать LLM (локально или через API) чтобы определять тип документа и вытаскивать ключевые слова.»* Упомяните, что **процесс импорта асинхронный** – реализована очередь задач: файл сканируется в фоне, а UI получает обновления статуса[[13]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L74-L81). У технарей это вызовет понимание, что система эффективно справляется даже с большим числом файлов (не зависает, а обрабатывает в несколько потоков). Можно упомянуть, что поддерживаются популярные форматы (PDF, DOCX, PNG, MP3, MP4 и т.д.), и что код для обработки расширяем – при желании можно добавить свои парсеры.

**3. Классический поиск с фильтрами (примерно 2–3 мин):**

* **Говорим:** Теперь, когда библиотека наполнена и проиндексирована, **покажем поиск**. Начните с **классического поиска** – это поиск по ключевым словам, похожий на привычный поиск в документах, но умнее. Введите в строку поиска (вверху страницы каталога) какой-нибудь запрос, например тему, которая есть в ваших документах, или фамилию автора. **Agregator мгновенно найдёт все файлы, где это слово встречается в названии, авторе, ключевых словах или даже в тексте**[[14]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L35-L35). Обратите внимание: поиск учитывает формы слов (для русского – морфология). Скажите: *«Можно вводить слова в любом склонении, система сама приведет их к начальной форме. Например, по запросу “алгоритмов” найдутся документы со словом “алгоритм”.»* Кроме того, Agregator знает некоторые синонимы и аббревиатуры: поиск умный, найдет схожие термины в тексте (если включён «умный режим»).

Отметьте, что **поиск работает очень быстро**: за доли секунды, благодаря использованию встроенной SQLite-индексации (FTS5) по тексту документов[[15]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L102-L109). И результаты можно сразу фильтровать.

* **Показываем:** После ввода запроса покажите **список результатов**: он обновляется прямо под строкой поиска. Расскажите, что можно уточнить результаты с помощью **фильтров**: в интерфейсе, как правило, есть фильтры по типу материала, году, авторам, коллекциям, тегам. Покажите: *«Допустим, слишком много результатов. Давайте отфильтруем: выберем тип “Статьи” и год “2021–2025”.»* Кликните на фильтры (например, выпадающий список годов или поле авторов – если оно есть). Вы увидите, как список сузится. Это **фасетный поиск**: он позволяет комбинировать условия (например, показать только презентации по теме X за последние 5 лет). Обратите внимание на блок с тегами: может быть, в интерфейсе есть облако тегов или список наиболее встречающихся ключевых слов[[16]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L91-L94). Если да – покажите, что можно кликнуть по тегу, и тоже будет фильтрация.

Итог: пользователь может найти иголку в стоге сена за секунды, даже если коллекция огромна.

* **A)** *Упрощённо:* Поясните, что **это похоже на поиск в Google, только по вашим материалам**. Акцентируйте удобство: *«Больше не нужно перебирать папки – достаточно вспомнить пару слов из названия статьи или имя автора.»* Для гуманитарной аудитории можно показать поиск по автору (например, «Толстой» – найдутся все материалы, где в авторах Толстой) или по теме («Вторая мировая» – всплывут все лекции на эту тему). Упомяните: *«Можно сочетать фильтры, как в интернет-магазине: например, показать только книги 2020 года по философии.»* Не углубляйтесь в технические нюансы, главное – **скорость и простота** этого поиска.
* **B)** *Технически:* Здесь уместно упомянуть про **FTS5 (Full-Text Search) в SQLite** и морфологию. Скажите, что Agregator создает отдельные виртуальные таблицы для быстрого поиска по тексту[[4]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L61-L68), поддерживает русский язык через библиотеку Pymorphy2 (лемматизация)[[17]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L104-L109), и кеширует результаты популярных запросов для мгновенной выдачи[[17]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L104-L109). Для фильтров – подчеркните, что они накладываются на SQL-запрос (например, *WHERE year BETWEEN X AND Y*) и также используют индексы. Если аудитория понимает, можно кратко упомянуть, что **поиск происходит локально SQL-запросом с MATCH по FTS-индексу**, без обращения к внешним сервисам – отсюда скорость. Это покажет масштаб внутренней работы (серьёзно, как маленький поисковик внутри).

**4. Интеллектуальный AI-поиск и ответы на вопросы (примерно 3 мин):**

* **Говорим:** Классический поиск хорош, когда вы знаете ключевые слова. Но что если у вас **вопрос** по материалам? Например, «Какой вывод сделан в статье X?» или «Объяснить понятие Y согласно лекциям». **Agregator оснащён AI-панелью, где можно задавать вопросы на естественном языке, и он попытается ответить, опираясь на загруженные документы**[[18]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L92-L94). Это как иметь собственного консультанта, который прочитал всю вашу библиотеку.

Объясните механизм понятными словами: *«Agregator сначала ищет релевантные документы, затем* *анализирует содержимое найденных файлов* *и формирует ответ в виде связного текста,* *добавляя ссылки на источники.»* То есть, он не «придумывает» от себя – в ответе каждая фраза подкреплена цитатой из ваших загруженных материалов. По сути, реализован **RAG (retrieval-augmented generation)**: вытаскиваем информацию и генерируем ответ. Это очень ценная функция для студентов и исследователей: можно быстро получить справку или выжимку по вопросу, не читая десятки статей вручную.

* **Показываем:** Переключитесь в интерфейсе на вкладку **«AI-панель»** (возможно, кнопка с иконкой или отдельная страница). Там обычно есть поле для ввода вопроса. Задайте пример: *«Например, спросим: “Что такое нейронная сеть?”»* – что-нибудь, о чём в ваших материалах точно есть информация. Нажмите «Поиск» или соответствующую кнопку. Покажите, что **начинается процесс AI-поиска**: скорее всего, на экране появляется индикатор или текст о ходе поиска (Agregator может выводить строки прогресса: например «Найдено 5 кандидатов... генерируется ответ...»). Через несколько секунд вы получите **сформулированный ответ** – несколько абзацев текста, где указано объяснение понятия и, возможно, упомянуты ключевые авторы или источники[[19]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L111-L115)[[20]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L120-L122). Главное – внизу или в самом ответе вы увидите **список источников**: ссылки на конкретные файлы и страницы, откуда взята информация.

Пролистайте ответ: поясните аудитории, что *«Вот Agregator нашёл ответ и сразу дает ссылки на документы, откуда он это взял. Можем кликнуть на источник и открыть первоисточник.»* Кликните на одну из ссылок – скорее всего, откроется встроенный просмотрщик PDF на нужной странице или предложит скачать. Это демонстрирует **прозрачность AI**: он не просто ответил, но и показал, **откуда** ответ получен.

Отметьте, что AI-поиск особенно полезен, когда нужно обобщить информацию из нескольких документов: *«Например, при подготовке курсовой вы можете спросить: “Какие методы описаны в источниках для решения задачи X?” и получите консолидированный ответ сразу с цитатами из разных работ.»*

* **A)** *Упрощённо:* Избегайте излишних технических терминов. Назовите AI-панель **«интеллектуальным ассистентом»**. Подчеркните, что он не заменяет чтение, но **очень экономит время**: *«Представьте, вам не нужно вручную перечитывать 10 статей – Agregator сам прочитает их и выдаст суть.»* В ответе всегда есть ссылки: скажите, что *«это как отчет с ссылками, по которым можно перейти и убедиться, что всё правильно»* – это важно для доверия. Гуманитариям можно сравнить с консультацией эксперта: *«словно вы спросили у преподавателя, а он не только ответил, но и дал список источников, где посмотреть детали.»*
* **B)** *Технически:* Здесь аудиторию можно впечатлить внутренней реализацией. Кратко опишите **многошаговый pipeline**: *«Agregator расширяет запрос с помощью LLM – под капотом используется, например, локальный Llama через LM Studio – затем ищет кандидатов через полнотекстовый поиск по базе, рассчитывает вес значимости (IDF) каждого термина, выбирает топ-15 документов,* *вычитывает из них фрагменты текста* *с совпадениями и отправляет эти фрагменты модели LLM для формирования связного ответа*[*[21]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md#L6-L9)[*[19]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L111-L115)*. При этом выполняется пост-обработка: находятся те предложения, где ответ, и они приводятся как цитаты.»* Упомяните, что **поддерживается кэширование** результатов ИИ (чтобы повторные схожие вопросы отвечались мгновенно)[[22]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L116-L118). Также отметьте, что Agregator может работать с разными LLM-сервисами – хоть с OpenAI API, хоть с локальным сервером LM Studio[[23]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L124-L128), так что его AI-функции гибко настраиваются. Важно: подчеркните, что **всё происходит локально** (если подключен локальный LLM), или на выбранном пользователем сервере – никакой отправки данных без ведома пользователя. Это наверняка вызовет интерес у IT-специалистов, т.к. показывает современность подхода (RAG, LLM, свои кеши, SSE-прогресс).

**5. Голосовой ввод и озвучивание ответов (примерно 1–2 мин):**

* **Говорим:** Ещё одна удобная функция – **голосовой поиск**. Иногда проще **сказать** вопрос, чем печатать, особенно на ходу или если руки заняты. Agregator позволяет нажать кнопку с микрофоном, **надиктовать вопрос голосом**, а система распознает речь и выполнит поиск по базе знаний[[20]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L120-L122). Также ответы, которые даёт AI, можно **прослушать вслух** – встроен синтез речи. Это полезно, например, чтобы слушать конспект лекций как аудиокнигу.
* **Показываем:** В AI-панели (или окне поиска) найдите значок микрофона – нажмите его. Скажите в микрофон какой-нибудь запрос (например: *«Опиши архитектуру программы Agregator»* – если уверены, что материалы это покрывают, или любой вопрос по вашим данным). Отпустите кнопку – система распознает речь (может появиться текст вашего вопроса автоматически в поле). Потом автоматически запустится поиск. Покажите, что это работает: *«Мы задали вопрос голосом – Agregator его* *распознал текст* *через встроенную модель и сразу начал поиск.»* Это особенно впечатлит, если в зале есть те, кому лень печатать или кто любит голосовых ассистентов.

Далее, когда получен ответ, продемонстрируйте **озвучивание**: возможно, рядом с ответом есть кнопка «Прослушать» или значок динамика. Нажмите его – вы услышите синтезированный голос, зачитывающий текст ответа. Скажите: *«Agregator может и проговорить ответ – удобно, когда, скажем, едешь в машине и слушаешь материалы.»* Упомяните, что можно регулировать скорость воспроизведения, голос, и что используется стандартный Web Speech API для озвучки[[20]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L120-L122).

* **A)** *Упрощённо:* Фокус на удобстве: *«Даже если вы устали печатать – просто спросите голосом. Это как Siri или Алиса, но умнее, потому что отвечает именно по вашим материалам.»* Отметьте, что для студентов, которые привыкли использовать голосовые сообщения, такая функция делает работу с системой более привычной и доступной.
* **B)** *Технически:* Упомяните, что **распознавание речи происходит локально** (если настроен Faster Whisper как бэкенд)[[20]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L120-L122), либо можно подключить внешний сервис. Формат данных – аудио от браузера (Web API) отправляется на /api/voice-search, где Agregator обрабатывает его. Про озвучку: используется Web Speech API в браузере для синтеза – т.е. здесь особо код не задействован, но можно сказать, что **поддержаны разные языки и голосовые движки** (в зависимости от ОС). Технарям будет достаточно знать, что эта возможность есть «из коробки» и её легко включить.

**6. Граф связей между авторами, тегами и темами (примерно 2 мин):**

* **Говорим:** Далее, **визуализация знаний**. Agregator строит **граф связей**, который позволяет увидеть, как ваши материалы связаны между собой по авторам, тегам и темам[[24]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L93-L94). Это что-то вроде **интеллект-карты вашей библиотеки**. Зачем это нужно? Например, чтобы выявить основные темы, над которыми вы работаете, посмотреть, какие авторы чаще всего встречаются, как они пересекаются с темами. Для исследователей это способ увидеть «картину в целом» по своей базе знаний.
* **Показываем:** Перейдите на страницу **«Граф»** или **«Связи»** (в интерфейсе, судя по документации, такой раздел есть). Вы увидите сетевую диаграмму: узлы – это авторы, теги или материалы, а связи показывают совместные встречаемости. Возможно, в интерфейсе можно выбрать режим: граф или облако тегов[[24]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L93-L94). Попробуйте, например, переключить на «облако тегов» – тогда система покажет просто самые частые ключевые слова крупнее, а редкие мельче. Затем вернитесь к режиму графа. Найдите на графе интересный узел (например, тег «Machine Learning») – кликните на него. Скорее всего, подсветятся все связанные узлы: например, авторы, у которых есть работы по Machine Learning, и названия соответствующих статей. Покажите: *«Видим, что у нас много материалов по Machine Learning – связанные авторы и темы выделены. Можно нажать на автора – и сразу получить список его работ в нашей базе.»* Обычно при клике на узел графа есть опция перейти к фильтрованному списку материалов.

Обратите внимание на возможности: граф можно перетаскивать, масштабировать. Если он слишком загроможден, возможно, есть фильтры (по коллекциям или по типам узлов). Упомяните, что **не каждый знает, что у него столько взаимосвязей между материалами – граф наглядно это показывает**. Это хорошо работает для междисциплинарных коллекций, где можно визуально выявить пересечения тематик.

* **A)** *Упрощённо:* Сравните граф знаний с **умственной картой**: *«Представьте, ваша библиотека – это вселенная, а граф показывает, какие “планеты” (темы, авторы) образуют созвездия.»* Скажите, что даже **без специальных знаний** можно интуитивно понять, какие темы наиболее широко охвачены. Для преподавателей-гуманитариев, например, это способ визуально показать студентам, какие области литературы они закрыли материалами, а где пробелы.
* **B)** *Технически:* Отметьте, что граф строится на основе данных в базе: например, узлы «автор» связаны с узлами «тег», если автор писал работу по этой теме. Можно упомянуть, что Agregator помечает некоторые теги как **графовые** (в настройках коллекций есть флаг graphable) – обычно это авторы, темы, возможно, организации[[25]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md#L15-L19). **Визуализация реализована на d3.js** (или другой библиотеке) – технарям будет интересно, что можно интегрировать свои способы отображения. Также отметьте, что есть режим «облако тегов», который просто по частоте строится – это классика анализа текста. Если доступны цифры (например, топ-10 тегов), можно озвучить, чтобы дать конкретики: *«Вот самый частый тег – “ML”, он встречается в 15 материалах».*

**7. Статистика и мониторинг базы знаний (примерно 1–2 мин):**

* **Говорим:** Agregator также предоставляет **страницу статистики**, где автоматически собираются различные аналитические данные по вашей библиотеке[[16]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L91-L94). Например, сколько у вас материалов по годам (график распределения по годам издания), кто самые популярные авторы в коллекции (диаграмма), распределение по типам материалов (сколько статей, сколько презентаций, сколько изображений), наиболее частые теги, средний объём документов и т.п. Это полезно, чтобы понимать состав вашей базы знаний и следить за её ростом.
* **Показываем:** Откройте вкладку **«Статистика»**. Покажите аудитории пару графиков: *«Вот, например, распределение наших документов по годам – видно, что большинство материалов свежие, за последние 5 лет.»* или *«А здесь – облако авторов: крупнейшие имена – это те, чьих работ у нас больше всего.»* Если есть график по типам, отметьте: *«У нас, допустим, 60% – статьи, 30% – книги, 10% – лекционные материалы.»* Также, если есть показатели, вроде *количество материалов в базе*, *средний размер файла*, *сколько документов без тегов* (возможный индикатор заполненности метаданных) – упомяните их. Цель – показать, что **система автоматически собирает отчёты**, и вам не нужно всё это считать вручную.

Для преподавателей и администраций учебных заведений это может быть интересным инструментом мониторинга: видно, как пополняется библиотека, какие темы преобладают, каких материалов не хватает.

* **A)** *Упрощённо:* Не погружаясь в цифры, скажите: *«Статистика нужна, чтобы* *видеть прогресс: например, за год вы добавили 100 новых статей – рост базы налицо! Или можно показать студентам, что материалы охватывают такие-то годы, таких-то авторов – это придаёт наглядность вашей работе.»* То есть, подчеркните пользу: **лучше понимать свою коллекцию**. Избегайте специфических терминов (дата-сеты, агрегация), говорите проще: «считает и рисует графики».
* **B)** *Технически:* Можно упомянуть, что статистика собирается динамически из базы (SQL-агрегаты) или в кэше, и обновляется при каждом сканировании. Например: *«Для быстрого доступа Agregator кеширует агрегированные данные (количество по годам, тегам и т.д.)*[*[26]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L62-L68) *– так графики строятся мгновенно.»* Технарям можно отметить, что эти данные также доступны через API, и, например, **можно выгрузить список всех тегов или получить CSV с каталогом** для внешнего анализа. Также упомяните существование **админ-панели** для мониторинга (аптайм, состояние очереди задач, логи)[[27]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L8-L11) – IT-специалисты ценят наличие средств контроля системы.

**8. Офлайн-редактор** AiWord **для написания текстов (примерно 2 мин):**

* **Говорим:** Отдельно стоит упомянуть уникальную функцию Agregator – встроенный редактор **AiWord** для подготовки научных текстов. Это офлайн-инструмент, который интегрирован с вашей библиотекой знаний. Чем он полезен? Когда вы пишете, скажем, статью, диплом или отчёт, AiWord позволяет **использовать материалы из Agregator прямо в процессе написания**. Он может подсказывать релевантные источники, автоматически оформлять библиографию (BibTeX), и даже помогать с черновиками текста с помощью локального ИИ.
* **Показываем:** Скорее всего, AiWord открывается по адресу http://localhost:5050/app/aiword или через меню. Перейдите в **AiWord**. Интерфейс похож на упрощённый текстовый редактор (похожий на Google Docs или Word, но с панелью помощника). Расскажите на словах возможности, демонстрируя на экране:
* **Синхронизация библиографии:** AiWord автоматически подтягивает список ваших источников (все материалы Agregator) в виде библиографической базы (BibTeX)[[28]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L44-L46)[[29]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L49-L52). Когда вы цитируете что-то, он может оформить ссылку.
* **Генерация плана и текста:** Покажите кнопки или меню, если есть, для генерации. Объясните: *«Вы можете попросить сгенерировать план статьи или черновик параграфа – AiWord использует локальный LLM, подключенный к Agregator, чтобы помочь вам написать текст»*[[30]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53). Например, нажмите «Сгенерировать план» – подождите и появится примерный план работы. Это демонстрирует, как AI может ускорить подготовку текста.
* **Вставка цитат из базы знаний:** Введите в тексте какое-то утверждение, а затем покажите функцию **вставки источников**. Например, выделите фразу и нажмите «Добавить источник» – AiWord выполнит поиск по библиотеке Agregator и вставит выдержку с ссылкой прямо в текст[[30]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53). Объясните: *«AiWord интегрирован с нашим AI-поиском: он может по ходу текста найти нужную цитату из ваших материалов и сразу добавить её с оформлением.»* В разделе «Источники каталога» (если есть внизу редактора) отобразятся все материалы, которые вы цитировали, чтобы потом легко сделать список литературы.

Подчеркните, что **AiWord работает офлайн** и создан специально для научных работ: он упрощает рутинные задачи написания – от генерации идей до правильного цитирования источников[[28]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L44-L46).

* **A)** *Упрощённо:* Сравните AiWord с обычным текстовым редактором, но «настероидах». *«Вместо того, чтобы бесконечно переключаться между вашим текстом и папкой с PDF-ами, вы можете прямо во время написания попросить систему: “найди мне цитату про методику X” – и она найдет в вашей библиотеке нужное и вставит.»* Это особенно полезно студентам, пишущим рефераты или дипломы: *«Экономит часы работы на оформление ссылок и поиск по источникам.»* Добавьте, что **не нужен интернет**: обычно студенты ищут материал в сети, копируют – а тут все внутри, надежно и без риска подтянуть чужой контент извне (работаем только с тем, что вы сами собрали).
* **B)** *Технически:* Упомяните, что AiWord – это **SPA-приложение на React + Tailwind**, которое запускается вместе с Agregator[[28]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L44-L46). Оно общается с бэкендом через API (для BibTeX, для поиска)[[30]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53). Технарям можно сказать, что AiWord поддерживает генерацию текста через подключённую LLM (например, LM Studio) – по сути, это редактор с AI-помощником, работающим **без облака**. Отметьте, что есть **скрипт экспорта**, позволяющий собрать AiWord как автономное приложение[[31]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53) – преподаватели информатики оценят возможность, например, дать студентам автономный редактор с материалами курса. Также подчеркните: вся библиографическая информация сохраняется и синхронизируется, формируется BibTeX – значит, при желании можно сразу получить готовый список литературы. Это покажет глубину интеграции системы для специалистов.

**9. Система пользователей, ролей и совместной работы (примерно 1 мин):**

* **Говорим:** Наконец, если речь о командной работе (например, в научной группе или в университете), **Agregator поддерживает несколько пользователей с разными правами**. Можно завести аккаунты для преподавателей, студентов-ассистентов, исследователей – и настроить, кто что видит и может делать. **Роли доступа** гибкие: *viewer* (только просмотр), *editor* (может добавлять и редактировать материалы) и *owner/admin* (полный доступ)[[7]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L2-L2). К каждой коллекции можно применять ACL – списки, кто имеет доступ. Таким образом, вы можете, например, иметь приватную коллекцию “Мой диссертационный материал” доступную только вам, и общую коллекцию “Материалы курса”, куда имеют доступ студенты.

Также в системе ведётся **журнал действий**: видно, кто и когда добавил файл или отредактировал тег. Это помогает следить за изменениями, особенно полезно преподавателям, курирующим работу студентов.

* **Показываем:** Откройте раздел **«Администрирование»** (если есть в меню). Там вы увидите управление пользователями и ролями[[27]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L8-L11). Покажите список пользователей (возможно, у вас только один админ – себя – и можно упомянуть, что можно добавить других). Демонстрировать создание пользователя можно, если позволяет время: нажмите «Добавить пользователя», впишите имя, задайте роль. Обратите внимание, что также на админ-панели могут быть **настройки системы** (например, подключение LLM, сервисов) и **состояние системы** (количество фоновых задач, загрузка)[[27]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L8-L11) – быстро скажите, что *«тут можно посмотреть журналы изменений и состояние системы, но в детали уходить не будем»*. Основная мысль – **возможность совместной работы и контроль**.
* **A)** *Упрощённо:* Скажем, для студентов-гуманитариев можно просто отметить: *«Если вы работаете в паре или в группе, вы можете вместе пользоваться Agregator: у каждого будет свой аккаунт. Так материалы общего проекта всегда синхронизированы, и вы не перезапишете чужую работу случайно.»* То есть, упростить до понятия «несколько пользователей могут безопасно работать с одной базой». Не нужно вдаваться в ACL, просто **подчеркните возможность совместной работы**.
* **B)** *Технически:* Здесь можно упомянуть детали реализации безопасности: *«Аутентификация в системе стандартная (логин/пароль), права хранятся в базе, у коллекций есть таблица соответствия пользователей и прав доступа*[*[32]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L69-L72)*. Все API-запросы фильтруются по правам, так что один пользователь не увидит чужие закрытые материалы*[*[25]*](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md#L15-L19)*.»* Это сигнал, что система продуманна на уровне Enterprise-функционала. Также можно добавить: *«Есть смена настроек в UI, удобное управление ролями – все как в профессиональных приложениях.»* Это даст уверенность IT-преподавателям, что Agregator можно использовать в учебной практике с разделением доступа для студентов.

**Переход:** После демонстрации основных функций, сделайте паузу и спросите, все ли ясно, возможно ответьте на короткие вопросы. Затем перейдите к примерам использования, чтобы связать функции с конкретными сценариями аудитории.

## Примеры использования в обучении и науке (23–28 мин)

Теперь, когда аудитория увидела, *что умеет Agregator*, свяжем это с их непосредственными потребностями. Рассмотрите по очереди три типичных роли: **студент**, **преподаватель**, **исследователь** – и покажите, как каждый из них может использовать Agregator в своей работе.

**Сценарий 1: Студент готовится к докладу или курсовой работе.**

Представьте студента, который за семестр собрал много материалов по своей теме. Обычно перед написанием доклада наступает хаос: нужно вспомнить, где какая статья, быстро освежить их содержание, выписать цитаты. С **Agregator** этот процесс гораздо проще: - Студент загружает **все статьи, конспекты лекций, презентации** по теме в Agregator (если они еще не были загружены в течение семестра). - Использует **классический поиск** по ключевым словам, чтобы найти в материалах нужные места. Например, готовясь к докладу о климатических изменениях, ищет «парниковые газы» и сразу получает список всех упоминаний этого термина в своих материалах. - Затем открывает **AI-панель** и задаёт вопрос: *«Какой основной вывод в последних исследованиях по парниковым газам?»* Agregator выдаёт связанный ответ с цитатами из загруженных статей. Это помогает быстро составить обзор, убедиться, что ничего не упущено. - Студент может даже **надиктовать** вопрос голосом, если ему так удобнее, и прослушать ответ – например, повторяя материал на ходу, как аудио. - Переходит в **AiWord** (или заранее пишет в нём доклад): там с помощью генерации черновика получает структуру доклада (введение, основные пункты, заключение). Затем с помощью функции вставки источников добавляет в текст несколько цитат из статей через AI-поиск[[30]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53). В итоге у него черновик доклада уже с оформленными ссылками на источники. - Наконец, он смотрит **граф связей** или **статистику** по своей теме внутри Agregator, чтобы убедиться, что охватил основные аспекты (например, видит, что в материалах часто упоминаются два ключевых автора – проверяет, учтены ли их идеи в докладе).

**Что подчеркнуть:** Agregator экономит студенту время и силы. Вместо того чтобы вручную шерстить десятки документов, он получает ответы и наглядные связи. Кроме того, **студент учится работать с источниками правильно** – у него всё цитируется с указанием первоисточника, что ценят преподаватели.

*(Если есть время и возможность, можно показать мини-демо: например, реально выполнить поиск и AI-ответ по учебному вопросу или показать, как вставляется цитата в AiWord. Но это опционально, главное – описать историю.)*

**Сценарий 2: Преподаватель систематизирует материалы курсов и лекций.**

Теперь взгляд со стороны преподавателя. Допустим, у преподавателя накоплены **методички, презентации лекций, научные статьи** для нескольких разных курсов. Часто они лежат просто на жестком диске, и каждый семестр преподаватель тратит время, собирая нужные файлы, обновляя их, делясь со студентами. С Agregator можно сделать это эффективнее: - Преподаватель создает в Agregator **отдельные коллекции** для каждого курса: например, «История искусств – лекции», «История искусств – литература», «Общая теория – статьи» и т.д. Затем загружает туда все соответствующие материалы. Система всё просканирует: тексты лекций, изображения картин, аудиофайлы – всё будет проиндексировано. - Теперь у преподавателя **единая база**. Он может быстро искать по ней: например, перед лекцией повторить материал, набрав в поиске ключевое слово, – мгновенно найдёт нужный слайд или цитату из статьи. - Во время подготовки к занятию преподаватель спрашивает у AI-панели: *«Напомни, какие основные тезисы были в прошлой лекции по Ренессансу?»* Получает краткое резюме с ссылками на свою же лекцию и источник. Это помогает освежить память (ведь курсов может быть несколько, и детали забываются). - Он также видит через **статистику**, что, например, по курсу «История искусств» у него 80 материалов, основные авторы – Гомер, Платон (если это курс античности, условно) – это даёт уверенность, что материалов достаточно. Граф связей может показать, какие темы чаще всего фигурируют – например, «Барокко» сильно связано с «Картинами» – значит, можно сделать акцент. - Если преподаватель хочет **поделиться материалами со студентами**, он может либо дать им доступ к определённой коллекции (например, завести общего пользователя-viewer для класса), либо выгрузить подборку: в Agregator есть функция **экспорта** (можно сохранить материалы с метаданными в ZIP или BibTeX)[[33]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L38-L44). В крайнем случае, он может просто использовать Agregator как свой вспомогательный инструмент: на паре, когда студент спрашивает что-то, преподаватель быстро ищет ответ через AI-поиск по своей базе и отвечает более точно, ссылаясь на источник.

**Что подчеркнуть:** Преподаватель благодаря Agregator **экономит время на подготовку** (всё под рукой, поиск мгновенный) и **улучшает качество преподавания** (ничего не забыто, всегда есть точные данные из источников). Кроме того, **организация материалов** – он фактически строит цифровой архив своего курса, который можно легко актуализировать и пополнять.

*(Здесь можно вставить ремарку: для преподавателей информатики особенно привлекательно, что это open-source система – можно настроить под себя, установить на кафедре свой сервер с Agregator для всех преподавателей, и не зависеть от сторонних сервисов.)*

**Сценарий 3: Исследователь ведёт архив научной работы.**

Для научного сотрудника или аспиранта Agregator – почти необходимый инструмент. Допустим, исследователь собирает литературу для обзора по своей теме, а также хранит результаты экспериментов, отчёты, презентации конференций: - Он использует Agregator как **личную библиотеку научных статей**. Загружает PDF всех скачанных статей. Система распознаёт их, вытягивает абстракты, ключевые слова, автоматом помечает тегами (например, по названиям методов). - Теперь, когда нужно написать обзор литературы, исследователь **задаёт AI-поиску вопросы**: *«Какие подходы используются для X по данным моих источников?»* Agregator выдаёт консолидированный ответ с указанием, в каких статьях что говорится. Это помогает увидеть, **какие работы поддерживают какую идею**, и не пропустить важные публикации. - С помощью **AiWord** исследователь пишет черновик статьи, а Agregator подтягивает цитаты: *«Например, в работе [Иванов 2022] говорится то-то…»* – эту фразу можно сгенерировать или проверить через AI-панель, получить цитату, сразу оформить ссылку. В итоге, когда статья готова, у него уже есть готовый список литературы, экспортый из Agregator, в нужном формате. - **Граф связей** для исследователя – способ увидеть **социальную сеть знаний**: например, может выясниться, что два ключевых автора в его библиотеке часто упоминаются вместе – возможно, они соавторы или один ссылается на другого. Это подсказывает, на кого обратить внимание. Граф тегов покажет, какие темы доминируют – возможно, стоит поискать материалы по недостаточно представленным темам. - Также исследователь может поддерживать **актуальность базы**: например, настроив Agregator на папку, куда автоматически складываются новые статьи (или подключив папку синхронизации с Zotero/Mendeley). Тогда периодически он запускает сканирование – новые файлы добавляются. Через **журналы и статистику** он видит, сколько нового пришло, нет ли ошибок распознавания. - Если он работает в команде (например, над коллективным проектом), с помощью системы пользователей они обмениваются находками: каждый добавляет свои файлы, а все вместе могут искать по общей базе. Это предотвращает дублирование (не надо каждому хранить копию статьи – все пользуются одной).

**Что подчеркнуть:** Agregator становится для исследователя **персональным knowledge-менеджментом**. Он не только хранит PDF-ки, но и позволяет **извлекать из них знания быстро**. В условиях, когда каждая неделя появляется новая литература, такой инструмент даёт конкурентное преимущество – вы тратите меньше времени на рутину, больше на анализ результатов. И **всё под вашим контролем**: в любой момент можно выгрузить всю базу (есть резервное копирование в виде BibTeX, CSV, копии базы данных), ничего не потеряется.

*(Опционально, если в зале есть исследователи: можно упомянуть, что Agregator – отличный инструмент для проведения обзоров литературы или написания обзорных статей, где надо агрегировать много источников.)*

## Заключение (28–30 мин)

**Резюмируем основные тезисы** презентации: - Мы живём в эпоху информационного перенасыщения: у преподавателей, студентов, учёных копится масса цифровых материалов. Agregator приходит на помощь, предлагая **единое пространство для хранения и умной работы с знаниями**. - Система работает **локально, офлайн**, сохраняя приватность и контроль, но при этом предлагает **современные AI-инструменты** (поиск с пониманием, голосовой ввод, автоматическое резюме ответов, графы знаний). - Мы увидели, как Agregator **решает проблему хаоса в файлах**, экономит время и повышает эффективность работы с информацией. Будь то подготовка к занятию, написание статьи или исследовательский поиск – инструмент ускоряет и упрощает эти задачи. - Отдельно отметьте, что Agregator – **open-source проект**. Это значит, его можно бесплатно использовать, настраивать под себя, а также при желании участвовать в его развитии.

**Призыв к действию:** Пригласите аудиторию попробовать Agregator в деле. Скажите, что установить его достаточно просто (требуется Python или Docker – подробная инструкция есть). Покажите **ссылку на GitHub** проекта, откуда можно скачать код, прочитать документацию и инструкции: например, на слайде выведите: ***GitHub: igorryltsin-design/agregator*** или откройте страницу репозитория. Также упомяните, что там есть Wiki/документация (если есть) и сообщество, где можно задать вопросы.

Подчеркните: *«Agregator уже сегодня может стать вашим персональным ассистентом для работы с знаниями. Попробуйте его – и вы удивитесь, как вы раньше жили без него!»*

Поблагодарите за внимание. Предложите аудитории задать вопросы. Завершите презентацию на позитивной ноте, подчёркивая, что **Agregator – это шаг вперёд в организации учебной и научной работы**, объединяющий лучшее от библиотек, поисковиков и AI-технологий.

* **A)** *Упрощённо:* Сделайте акцент, что **каждый** – не только айтишник – сможет воспользоваться Agregator. Предложите студентам и преподавателям гуманитарных направлений не бояться технической стороны: *«Программа с понятным интерфейсом, на русском языке, с веб-интерфейсом – открываете как обычный сайт и работаете.»* Возможно, приведите контакты или сообщество, куда можно обратиться за помощью при установке. Заверьте, что попробовать стоит, ведь выгоды (сэкономленное время, порядок в материалах) того стоят.
* **B)** *Технически:* Отметьте, что проект развивается, и будет рад вкладу. *«Если у вас есть идеи или вы нашли баг – на GitHub можно открыть issue или даже внести свой код.»* Это покажет ИТ-специалистам и преподавателям информатики, что проект активный и открыт к сотрудничеству. Также можно упомянуть перспективы: *«В планах – интеграция с другими системами, улучшение модели графа, поддержка новых форматов»* – это может заинтересовать технарей поучаствовать. Завершите упоминанием, что **локальные инструменты, такие как Agregator, – это тренд** на импортонезависимость и контроль над данными, что особенно актуально для вузов и исследований.

Спасибо за внимание и **успешного освоения Agregator!**

[[1]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L1-L5) [[2]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L3-L6) [[3]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L42-L50) [[4]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L61-L68) [[5]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L33-L33) [[6]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L36-L37) [[7]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L2-L2) [[8]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L33-L36) [[9]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L84-L88) [[10]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L74-L79) [[11]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L83-L88) [[12]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L34-L35) [[13]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L74-L81) [[14]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L35-L35) [[15]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L102-L109) [[16]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L91-L94) [[17]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L104-L109) [[18]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L92-L94) [[19]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L111-L115) [[20]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L120-L122) [[22]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L116-L118) [[23]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L124-L128) [[24]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L93-L94) [[26]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L62-L68) [[27]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L8-L11) [[28]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L44-L46) [[29]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L49-L52) [[30]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53) [[31]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L50-L53) [[32]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L69-L72) [[33]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md#L38-L44) README.md

<https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/README.md>

[[21]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md#L6-L9) [[25]](https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md#L15-L19) readme\_ai.md

<https://github.com/igorryltsin-design/agregator/blob/58a147d2e390cd1c2d5b9fa35e80ba2f7978d688/docs/readme_ai.md>