ЦИФРОВОЙ ДНЕВНИК ЛАБОРАТОРИИ

(версия 1.0)

ТЕХНИЧЕСКОЕ ОПИСАНИЕ И ХАРАКТЕРИСТИКИ

Оглавление

1. ОБЩАЯ АРХИТЕКТУРА

Приложение представляет собой настольное приложение на Python с графическим интерфейсом на Tkinter. Основные компоненты:

- пользовательский интерфейс (GUI) – модульная структура в папке gui/;
- база данных (локальная, файл) – модуль database.py;
- модуль импорта данных спектрометра – spectrometer_importer.py;
- модуль генерации отчётов – report_generator.py;
- модуль парсинга отчётного текста и параметров – report_parser.py;
- модуль напоминаний для Windows – win_notifications.py;
- точка входа в приложение – main.py;
- модуль calendar_view.py – календарь и события;
- модуль experiment_editor.py – редактор экспериментов;
- модуль main_window.py – основная страница экспериментов;
- модуль main_menu.py – главное меню.

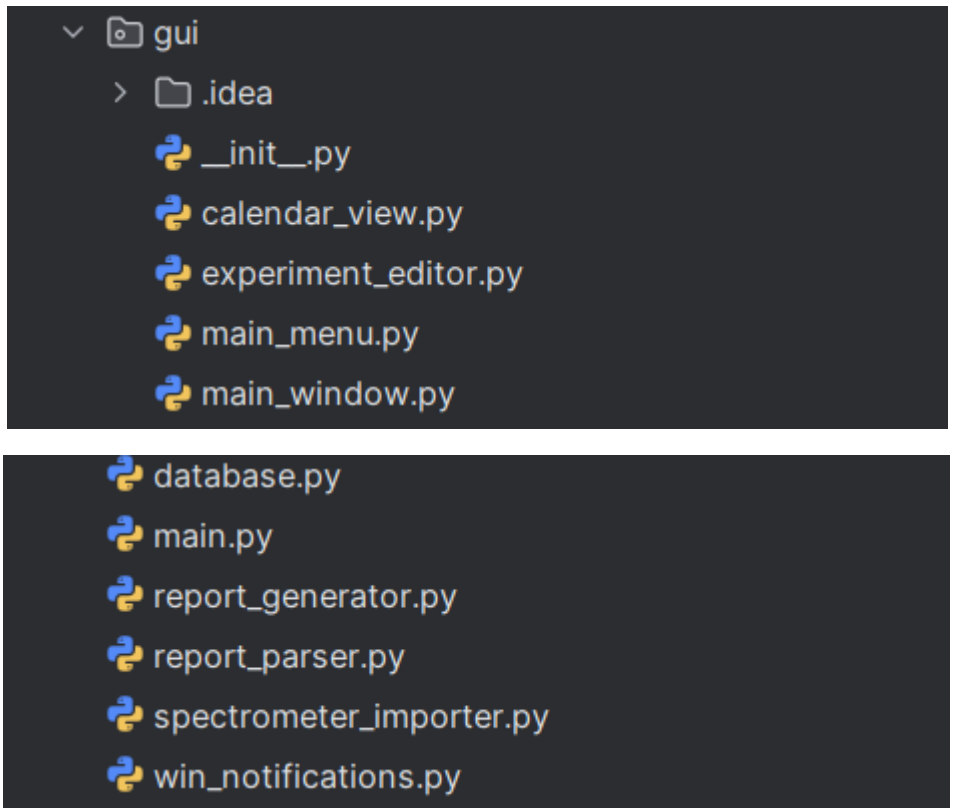


Рис. 1. Структура папок проекта (короткий обзор в файловом менеджере/IDE).

2. ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ И ЗАВИСИМОСТИ

Приложение использует язык программирования Python 3.x. GUI включает Tkinter (стандартная библиотека Python), ttk-виджеты и использование кастомных стилей (Style) для единообразного внешнего вида.

Графика и изображения формируются с помощью Matplotlib (backend 'TkAgg' в GUI, 'Agg' в генерации отчётов) и Pillow (PIL) для работы с иконками и фоновыми изображениями.

Отчёты генерируются с помощью reportlab для генерации PDF, верстки текста и таблиц, подключения TTF-шрифта для кириллицы, а также python-docx для формирования DOCX-отчётов (заголовки, абзацы, таблицы, вставка изображений).

База данных включает SQLite (через модуль database.py) для хранения экспериментов, данных спектрометра, вложений, событий календаря и напоминаний.

ОС и интеграция предполагают Windows 10+ и win_notifications.py для планирования и отмены напоминаний в виде нативных уведомлений Windows.

3. ОСНОВНЫЕ МОДУЛИ

3.1. main.py

main.py является точкой входа в приложение. Он инициализирует корневое окно Tk, создаёт экземпляр MainMenu (главное меню) и настраивает полноэкранный режим при запуске.

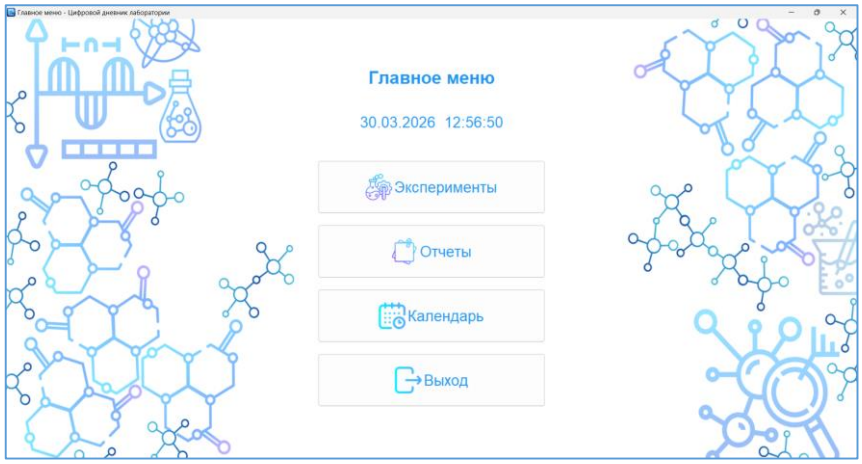


Рис. 2. Главное окно при запуске (см. также Руководство пользователя).

3.2. gui/main_menu.py

gui/main_menu.py содержит класс MainMenu, который отвечает за главное меню приложения и открывает следующие страницы: раздел экспериментов (MainWindow), страницу создания отчётов и страницу календаря. Работа осуществляется в «страничном» режиме: при переходе очищается содержимое root и создаётся новый экран в том же окне.

Функциональные элементы включают загрузку иконок из папки icons/, фоновое изображение главного меню (assets/main_window.png), динамическое отображение даты и времени (обновление каждую секунду) и стили для крупных кнопок (MenuButton.TButton).

Методы включают open_experiments() для открытия раздела экспериментов, open_reports() для открытия страницы создания отчёта (страница, не окно), open_calendar() для открытия календаря как страницы и exit_application() для корректного завершения работы.

Особенности страницы «Создание отчёта» включают выбор экспериментов через Treeview с чекбоксами, выбор формата отчёта (PDF/DOCX) через радиокнопки и белый фон под блоком формата и кнопками без серых панелей.

3.3. gui/main_window.py

gui/main_window.py содержит класс MainWindow, который представляет основную страницу «Эксперименты». Основные обязанности включают создание и настройку меню приложения (Файл / Отчёт / Вид / Справка), создание панели инструментов, организацию трёхколоночного layout (список, детали, спектр/вложения), работу со списком экспериментов и их фильтрацией, отображение деталей эксперимента, параметров, примечаний, отображение спектров и вложений, запуск генерации отчётов, переход к редактору эксперимента и календарю в режиме страниц.

Ключевые поля включают self.db (соединение с базой данных), self.importer (экземпляр SpectrometerImporter), self.report_gen (экземпляр ReportGenerator), self.current_exp_id (текущий выбранный эксперимент), self.current_x_axis и self.spectrum_color (настройки спектра: ось, цвет).

Важные методы включают new_experiment(), который открывает ExperimentEditor как страницу: закрывает текущее соединение self.db,

очищает root, создаёт отдельное соединение editor_db и ExperimentEditor(..., as_page=True, on_back=...).

- Метод edit_experiment() работает аналогично new_experiment, но с указанным experiment_id.
- Метод show_calendar() открывает CalendarView как страницу (as_page=True).
- Метод generate_report(format_type) собирает ID отмеченных экспериментов и вызывает ReportGenerator.generate_pdf/generate_docx.

Настройка спектра осуществляется через show_experiment_details(), который читает metadata эксперимента: spectrum_x_axis и spectrum_color, а update_spectrum_view() строит график спектра в правой части страницы.

Страница редактирования спектра edit_spectrum() переведена на модель страниц: закрывает текущее соединение self.db, очищает root, создаёт отдельное соединение spec_db, строит страницу с FigureCanvasTkAgg (Matplotlib), комбо для оси X и цвета, кнопками «Сохранить спектр» и «Назад», читает и обновляет metadata['spectrum_x_axis'] и metadata['spectrum_color'] в базе через spec_db.

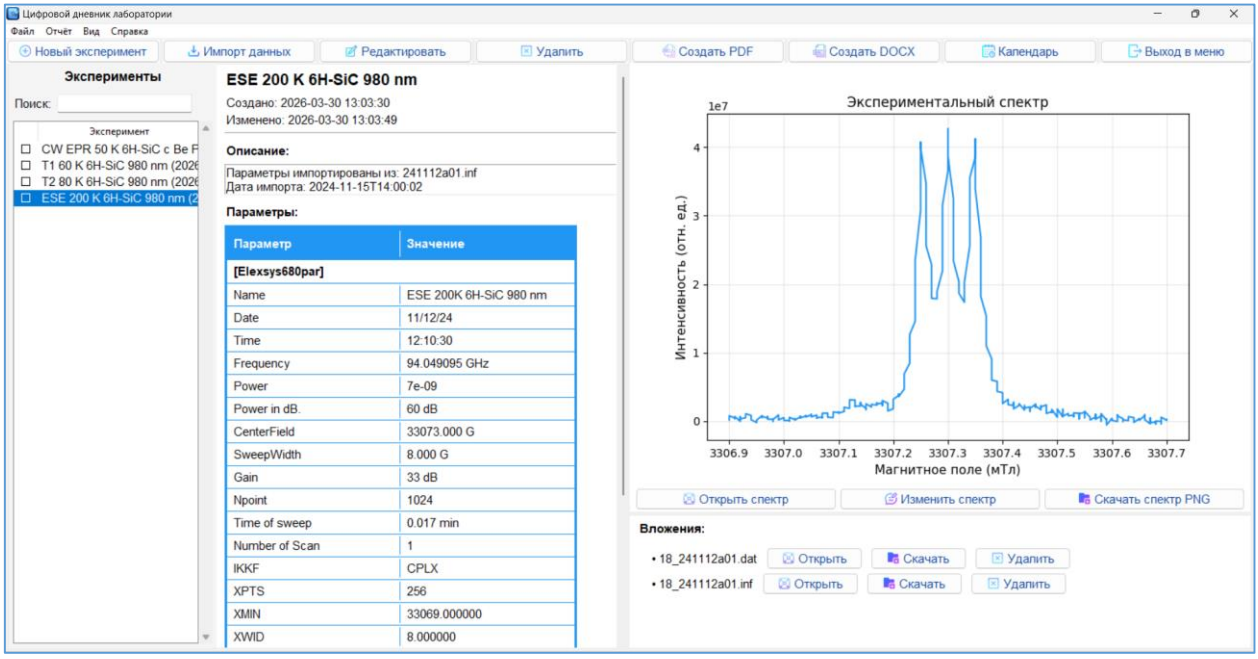


Рис. 3. Страница «Эксперименты» в разрезе.

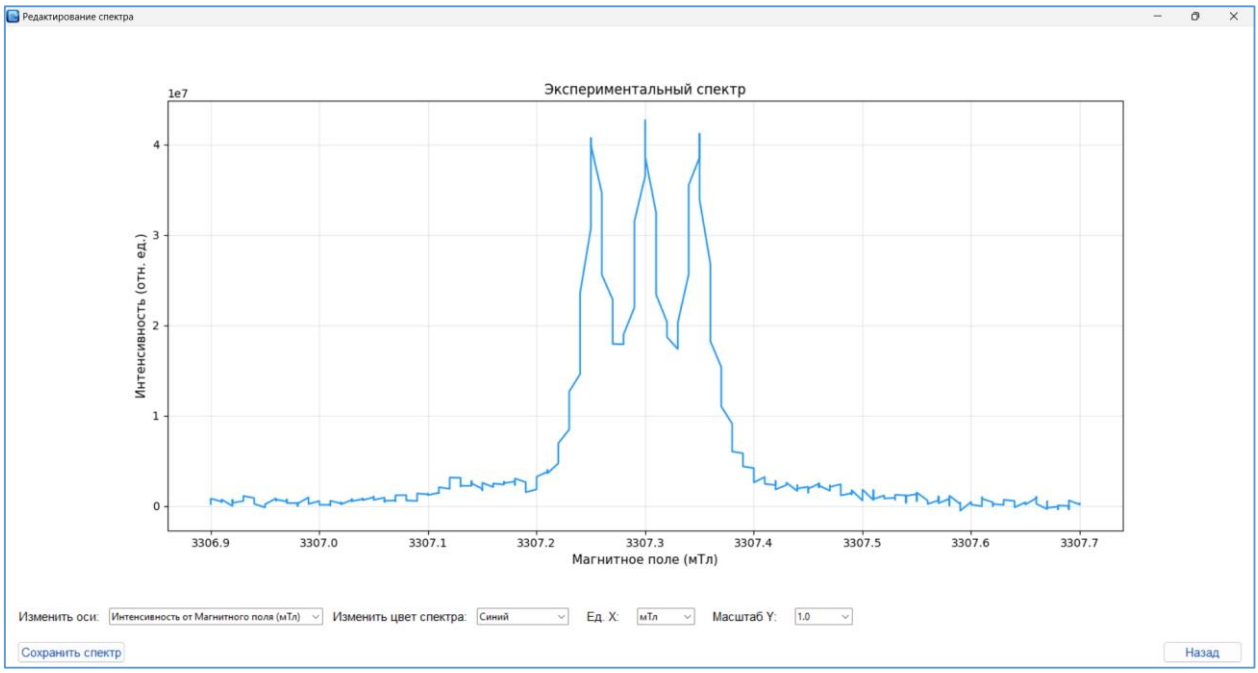


Рис. 4. Страница «Редактирование спектра».

3.4. gui/experiment_editor.py

gui/experiment_editor.py содержит класс ExperimentEditor, который представляет редактор экспериментов, работающий как отдельное окно (Toplevel) или как страница (режим as_page=True). Конструктор принимает parent (Tk или Toplevel), db (Database), experiment_id (None для нового эксперимента), as_page (True/False) и on_back (callback «Назад» в режиме страницы).

Основные функции включают редактирование полей:

- заголовок,
- дата эксперимента (метаданные),
- описание,
- параметры (в текстовом формате, включая импорт из INF),
- примечания,
- импорт/очистка данных INF (import_inf_parameters, clear_inf_parameters),
- импорт/удаление данных спектрометра (import_spectrometer_data, delete_spectrometer_data),

- работу с вложениями: добавление (копирование файлов в attachments с формированием имени), удаление из списка перед сохранением, сохранение в БД attachments.

Сохранение эксперимента (save_experiment) осуществляет создание или обновление записи в базе, запись/обновление поля metadata (например, experiment_date). Особенностью является то, что при сохранении вложения с временным именем temp_... переименовываются в файлы, привязанные к окончательному ID эксперимента.

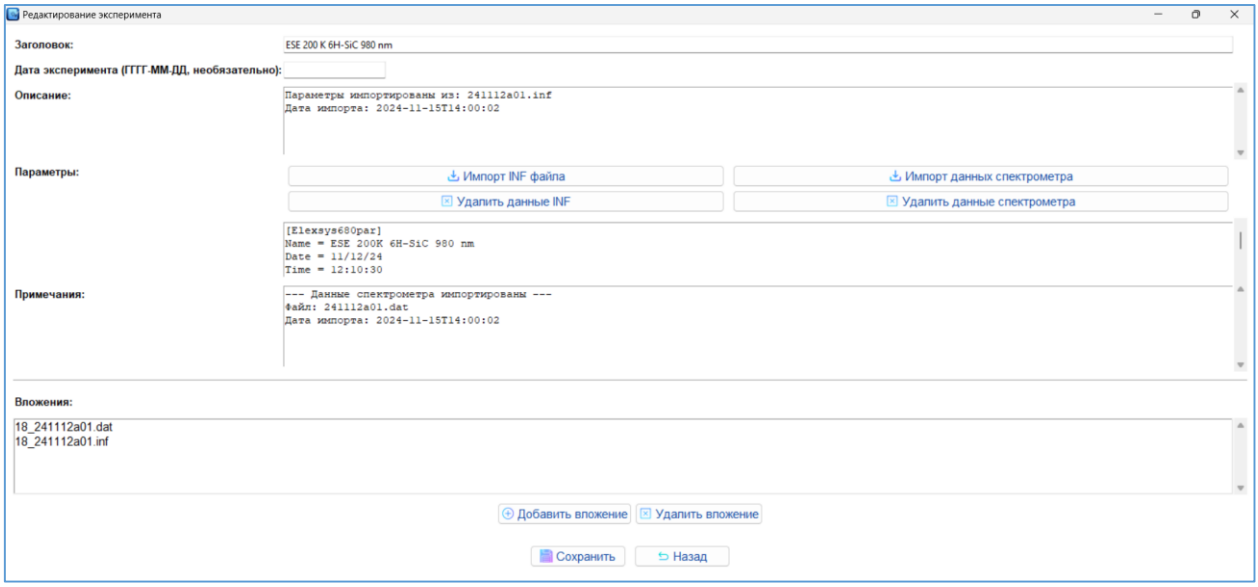


Рис. 5. Страница редактора эксперимента.

3.5. gui/calendar_view.py

gui/calendar_view.py содержит класс CalendarView для отображения календаря и списка событий. Режимы работы включают отдельное окно (Toplevel) и страницу (as_page=True), которая используется из главного меню и раздела «Эксперименты». Функциональность включает:

- навигацию по месяцам (предыдущий/следующий),
- отображение сетки дней с подсветкой текущей даты и индикаторами количества событий,
- список событий с колонкой чекбокса, датой/временем, названием, напоминанием,
- массовые операции: «Выбрать всё» (все чекбоксы), «Снять всё» (удаление всех событий), редактирование и удаление отмеченных

событий, интеграцию с Windows-уведомлениями (schedule/cancel).

Добавление/редактирование события использует класс CalendarEventDialog, который переведён на модель «диалог/страница»: при as_page=True использует parent как контейнер страницы и по кнопке «Назад» вызывает on_back; при обычном режиме – Toplevel с grab_set(). Варианты вызова включают CalendarView.new_event(): в режиме as_page закрывает db календаря, очищает окно, создаёт новое соединение для события, вызывает CalendarEventDialog(..., as_page=True, on_back=...); в старом режиме – открывает Toplevel и ждёт его закрытия. CalendarView.edit_event() работает аналогично new_event, но с существующим event_id.

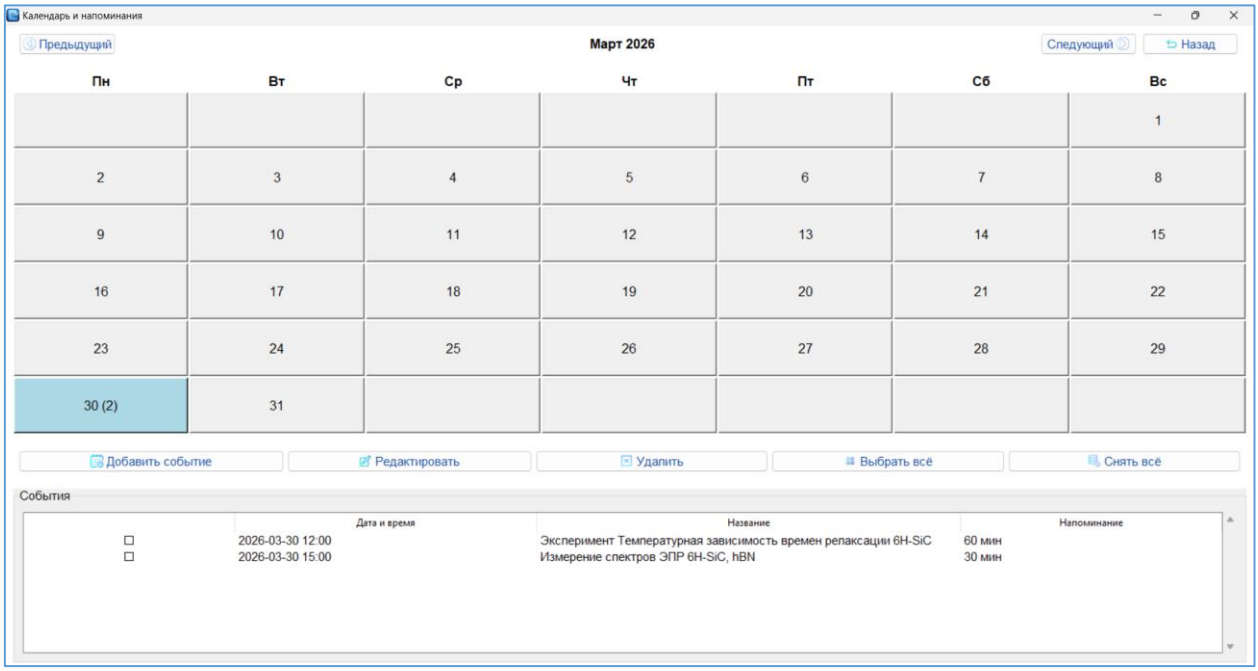


Рис. 6. Страница календаря.

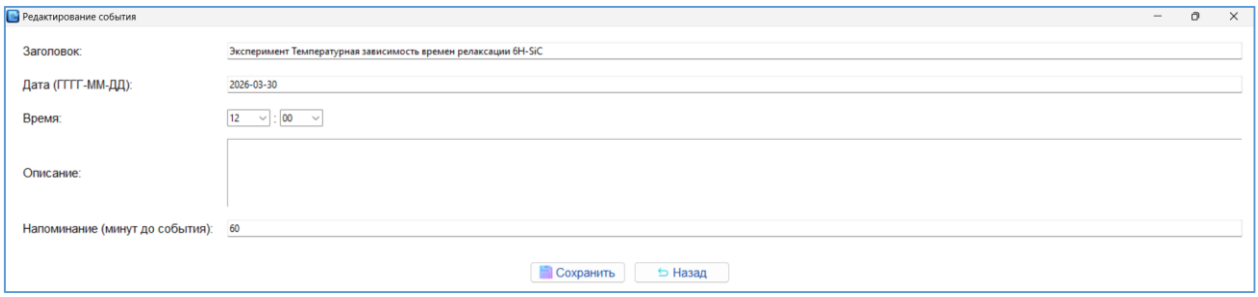


Рис. 7. Страница создания/редактирования события.

3.6. report_generator.py

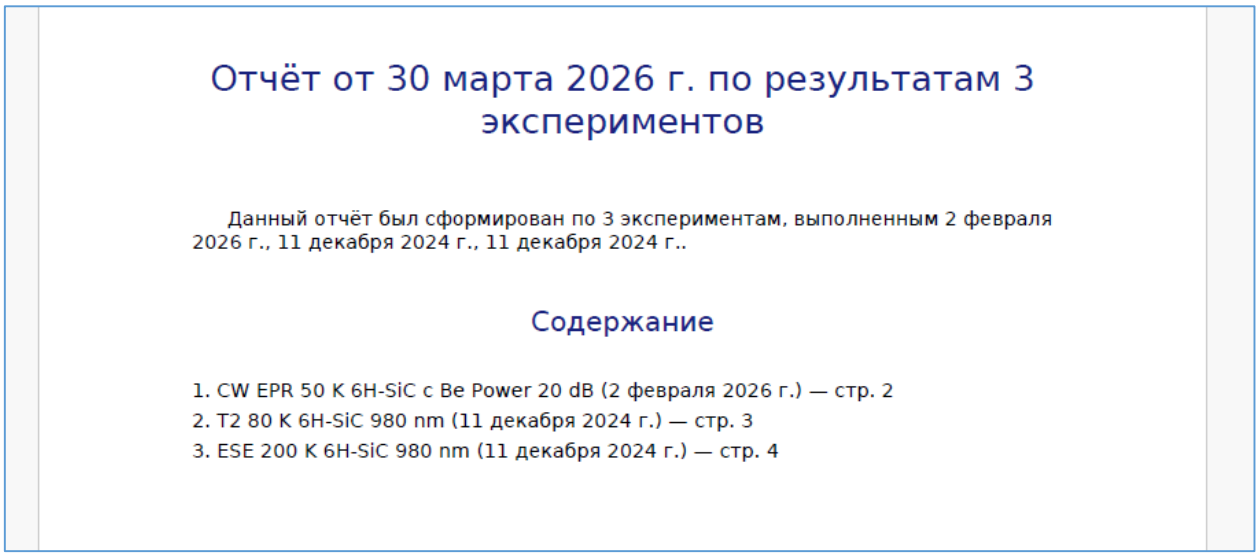
report_generator.py содержит класс ReportGenerator, который формирует отчёты в форматах PDF и DOCX. Методы включают generate_pdf(experiment_ids, output_path, current_x_axis='magnetic_field') и generate_docx(experiment_ids, output_path, current_x_axis='magnetic_field'). Общая схема работы:

1. по списку experiment_ids считывает записи экспериментов и связанные спектральные данные,
2. формирует контекст через build_report_context (report_parser.py), строит титульную страницу, содержание, отдельные «страницы» для каждого эксперимента: заголовок с датой, абзацы с описанием установки и параметров, таблицу параметров (4-колоночная), графики спектров (PNG, генерируемые Matplotlib'ом), финальный блок «Отчёт сформирован автоматически...».

PDF использует ReportLab: SimpleDocTemplate, Paragraph, Spacer, Table, Image, PageBreak, настраиваемые стили (заголовки, основной текст с выравниванием по ширине и отступом первой строки, таблицы параметров), подключение шрифта DejaVuSans.ttf из папки fonts/ для корректного отображения кириллицы, нумерацию страниц.

DOCX использует python-docx: заголовки разных уровней, абзацы с выравниванием по ширине и отступом первой строки, таблицы с синими границами, вставка графиков спектров как изображений, финальный блок в виде синих заголовков.

Построение графиков осуществляется через _create_spectrum_graph(spec_data, current_x_axis, spectrum_color, x_limits, y_limits): преобразует массив data_points в x/y, учитывает выбранную ось X, применяет цвет и пределы осей из metadata, сохраняет PNG во временную директорию, возвращает путь к временному файлу, по завершении отчёта временные файлы удаляются.

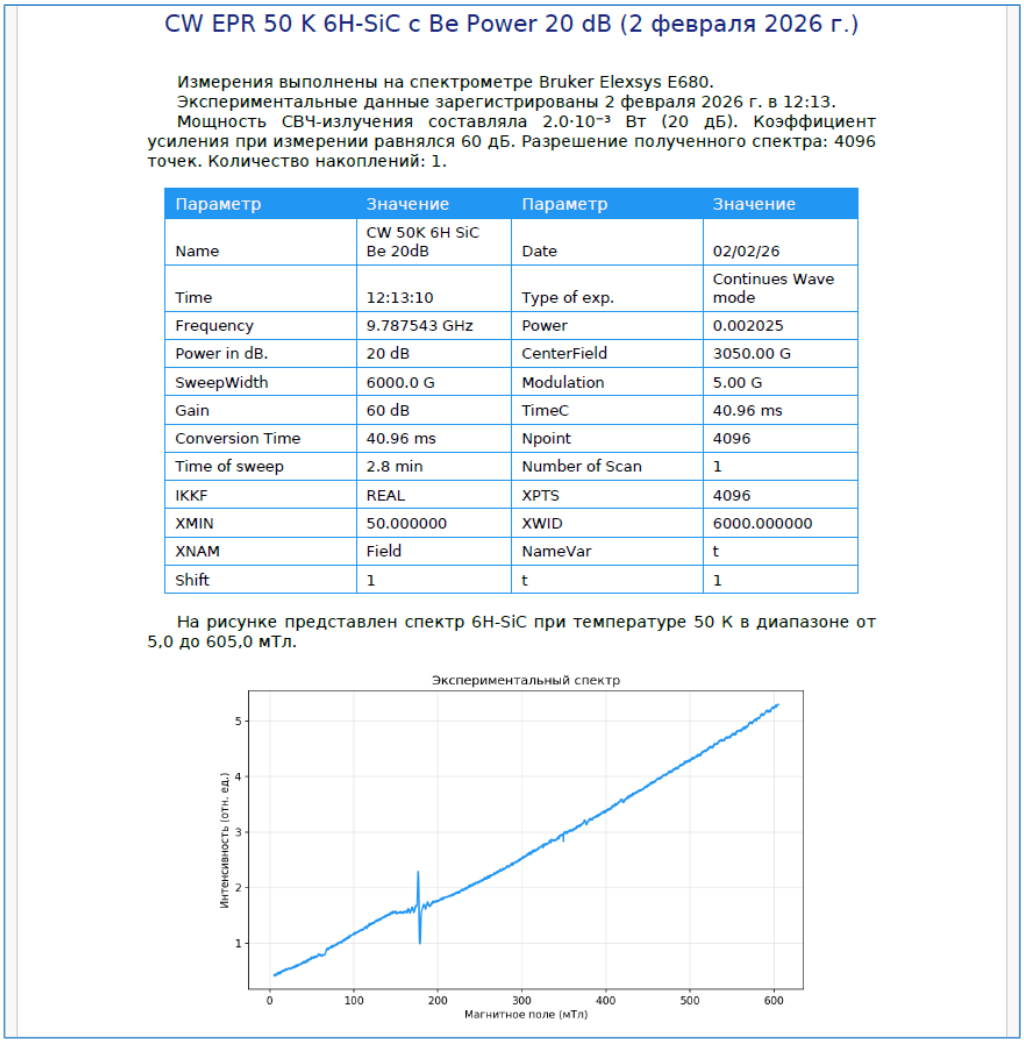


Отчёт от 30 марта 2026 г. по результатам 3 экспериментов

Данный отчёт был сформирован по 3 экспериментам, выполненным 2 февраля 2026 г., 11 декабря 2024 г., 11 декабря 2024 г..

Содержание

Рис. 8. Пример сгенерированного PDF-отчёта (титульная страница).



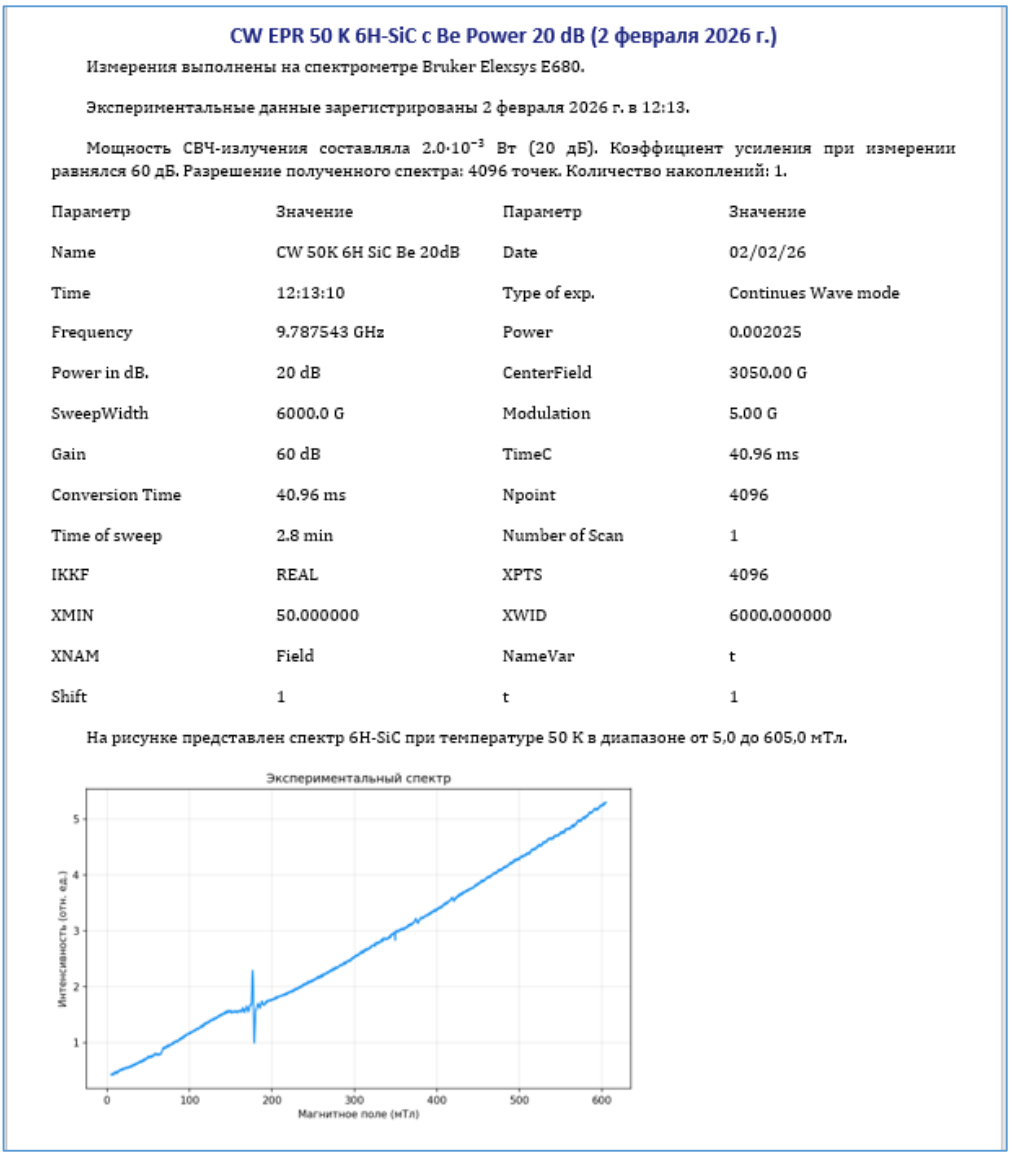
CW EPR 50 K 6H-SiC c Be Power 20 dB (2 февраля 2026 г.)

Измерения выполнены на спектрометре Bruker Elexsys E680.
Экспериментальные данные зарегистрированы 2 февраля 2026 г. в 12:13.
Мощность СВЧ-излучения составляла $2.0 \cdot 10^{-3}$ Вт (20 дБ). Коэффициент усиления при измерении равнялся 60 дБ. Разрешение полученного спектра: 4096 точек. Количество накоплений: 1.

Параметр	Значение	Параметр	Значение
Name	CW 50K 6H SiC Be 20dB	Date	02/02/26
Time	12:13:10	Type of exp.	Continues Wave mode
Frequency	9.787543 GHz	Power	0.002025
Power in dB.	20 dB	CenterField	3050.00 G
SweepWidth	6000.0 G	Modulation	5.00 G
Gain	60 dB	TimeC	40.96 ms
Conversion Time	40.96 ms	Npoint	4096
Time of sweep	2.8 min	Number of Scan	1
IKKF	REAL	XPTS	4096
XMIN	50.000000	XWID	6000.000000
XNAM	Field	NameVar	t
Shift	1	t	1

На рисунке представлен спектр 6H-SiC при температуре 50 K в диапазоне от 5,0 до 605,0 мТл.

Рис. 9. Страница эксперимента с таблицей параметров и графиком в PDF.



CW EPR 50 K 6H-SiC c Be Power 20 dB (2 февраля 2026 г.)

Измерения выполнены на спектрометре Bruker Elexsys E680.

Экспериментальные данные зарегистрированы 2 февраля 2026 г. в 12:13.

Мощность СВЧ-излучения составляла $2.0 \cdot 10^{-3}$ Вт (20 дБ). Коэффициент усиления при измерении равнялся 60 дБ. Разрешение полученного спектра: 4096 точек. Количество накоплений: 1.

Параметр	Значение	Параметр	Значение
Name	CW 50K 6H SiC Be 20dB	Date	02/02/26
Time	12:13:10	Type of exp.	Continues Wave mode
Frequency	9.787543 GHz	Power	0.002025
Power in dB.	20 dB	CenterField	3050.00 G
SweepWidth	6000.0 G	Modulation	5.00 G
Gain	60 dB	TimeC	40.96 ms
Conversion Time	40.96 ms	Npoint	4096
Time of sweep	2.8 min	Number of Scan	1
IKKF	REAL	XPTS	4096
XMIN	50.000000	XWID	6000.000000
XNAM	Field	NameVar	t
Shift	1	t	1

На рисунке представлен спектр 6H-SiC при температуре 50 K в диапазоне от 5,0 до 605,0 мТл.

Рис. 10. Аналогичная страница отчета в DOCX.

4. СХЕМА ДАННЫХ (ОБЩИЕ ПОЛЯ)

Подробная структура таблиц задаётся в database.py; здесь представлен логический обзор. Эксперименты включают: id (целое, первичный ключ), title (строка), description (текст), parameters (текст, в том числе ini-подобные строки и/или JSON-постфикс), notes (текст), metadata (JSON-строка: дата эксперимента, настройки спектра и т.п.), created_date (ISO-строка), modified_date (ISO-строка).

Спектральные данные включают: id, experiment_id, data_points (массив точек), x_axis_label, y_axis_label, дополнительные параметры (JSON или словарь), timestamp.

Вложения включают: id, experiment_id, file_path, file_type (расширение), description (имя вложения).

События календаря включают: id, title, event_date (ГГГГ-ММ-ДД), event_time (НН:ММ или пусто), description, reminder_minutes (целое, 0 = без напоминания), статус напоминания (если предусмотрен).

5. НАСТРОЙКА И СБОРКА

5.1. Зависимости

Необходимо установить следующие пакеты (имена могут отличаться в зависимости от окружения): reportlab, python-docx, matplotlib, pillow (PIL). Пример для pip (можно адаптировать под requirements.txt): reportlab, python-docx, matplotlib, pillow.

5.2. Запуск из исходников

Запуск из исходников:

1) Установить Python 3.x.
2) Установить зависимости.
3) Убедиться, что рабочая директория – корень проекта.
4) Запустить: python main.py

5.3. Сборка в .exe (Windows)

Сборка в .exe (Windows): рекомендованный инструмент – PyInstaller. Пример команды (упрощённо): указать основной модуль main.py, добавление папок assets/, icons/, fonts/, корректную установку backend'ов Matplotlib.

6. ПРИНЦИПЫ НАВИГАЦИИ (РЕЖИМ СТРАНИЦ)

Важное архитектурное решение – приложение работает в режиме страниц, а не всплывающих окон. Главный контейнер – одно окно Tk (root). При переходе на новый раздел дети root уничтожаются, создаётся новый

объект страницы (MainMenu, MainWindow, CalendarView, ExperimentEditor и т.п.), которому передаётся root. Возврат назад осуществляется через callback on_back, который закрывает (при необходимости) отдельное соединение с БД текущей страницы, очищает root, воссоздаёт предыдущую страницу. Исключения: старые режимы с Toplevel продолжают поддерживаться, но основной сценарий – страницы.

7. ОГРАНИЧЕНИЯ И РЕКОМЕНДАЦИИ

Приложение рассчитано на локальное использование одним пользователем. Совместный доступ к базе и параллельная запись из нескольких процессов не предусмотрены.

Рекомендуется:

- периодически делать резервные копии файла базы данных;
- не редактировать файлы базы вручную;
- хранить вложения и отчёты на надёжном носителе.

Возможные направления развития:

- расширение схемы metadata (дополнительные параметры, шкалы и т.п.);
- настройка диапазонов осей спектров (x_limits, y_limits) из GUI;
- многоязычный интерфейс (русский/английский);
- поддержка дополнительных форматов импорта/экспорта.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРИЛОЖЕНИЯ

1) Системные требования

- Тип приложения: настольное (desktop), локальное хранение данных.
- ОС (фактически целевая): Windows 10/11 x64 (сборка выполнялась под Windows 11).
- Процессор: x64, от 2 ядер, от ~1.6 ГГц (рекомендовано от 2.0 ГГц).
- ОЗУ: минимум 4 ГБ, рекомендовано 8 ГБ.
- Диск:

- для запуска: от 300–500 МБ свободного места;
- дополнительно под данные/вложения/отчеты – по объему работы.

- Графика: отдельная видеокарта не требуется (достаточно стандартной для GUI и построения графиков).

2) Технологический стек

- Язык: Python 3.x.
- GUI: tkinter (ttk).
- Графики спектров: matplotlib + numpy.
- Работа с изображениями/иконками: Pillow.
- Генерация отчетов:
  - PDF: reportlab;
  - DOCX: python-docx.
- База данных: SQLite (локальный файл experiments.db).
- Уведомления Windows: winrt, pywin32 (интеграция с Windows).

3) Зависимости и runtime

В разделе зависимостей и runtime-окружения указаны необходимые для работы приложения пакеты. Требуется установка библиотеки reportlab версии не ниже 4.0.0 и python-docx версии от 1.1.0. Для графических операций необходимы matplotlib версии >= 3.7.0, numpy версии >= 1.24.0 и Pillow версии >= 10.0.0. Интеграция с Windows обеспечивается пакетами winrt версии >= 1.0.21033.1 и pywin32 версии >= 306. Все компоненты должны быть совместимы с используемой версией Python и установлены в среду выполнения для корректного функционирования модулей генерации отчетов, обработки изображений и системных уведомлений. Дополнительно:

- Runtime разработки: Python 3.x.
- Runtime поставки: standalone .exe (через PyInstaller), Python на целевой машине не требуется.

4) Сетевые требования

- Интернет: не требуется для основной работы.
- Порты/протоколы: не используются (приложение локальное, без клиент-серверной архитектуры).
- Пропускная способность: не применимо.

5) Производительность

- Модель использования: однопользовательское локальное приложение.
- Отклик UI: интерактивный в рамках обычных объемов данных.
- Ограничения: зависит от объема спектральных точек, количества вложений и размера отчетов.
- Параллельные пользователи: не предусмотрены (локальный SQLite файл).

6) Безопасность

- Хранение данных: локально в SQLite (experiments.db), без облачной синхронизации.
- Аутентификация/авторизация: single-user desktop сценарий.
- Передача данных по сети: отсутствует по умолчанию.

7) Совместимость

- Поддерживаемые форматы импорта спектрометра: .txt, .dat, .inf.
- Форматы экспорта: .pdf, .docx, изображение спектра .png.
- Кодировка/язык: интерфейс и отчеты на русском, поддержка кириллицы (шрифт DejaVuSans для PDF).
- Интеграции: внешние API/веб-сервисы не требуются.

8) Развертывание и поставка

- Инструмент сборки: PyInstaller (build_exe.spec).
- Формат поставки: один исполняемый файл .exe + рабочая папка.
- Первый запуск: experiments.db создается автоматически при отсутствии файла.
- Установка: не требуется (portable-режим), запуск напрямую из папки.