**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра дифференциальных уравнений и системного анализа**

**Компьютерная программа в помощь методисту для занятий со слабослышащими детьми**

Курсовая работа

Шик Оксаны Сергеевны,

студентки 3 курса, специальность 1-31 03 09 Компьютерная математика   
и системный анализ

Научный руководитель:  
кандидат физ.-мат. наук,  
доцент А. Э. Малевич

Минск, 2024

Оглавление

Введение

В современном образовании особое внимание уделяется инклюзивному обучению и поддержке детей со слабым слухом. Однако, методистам, занимающимся обучением таких детей, может потребоваться дополнительная помощь и инструменты, чтобы обеспечить эффективное и доступное обучение.

В этом контексте разработка компьютерной программы, предназначенной специально для поддержки методистов при проведении занятий со слабослышащими детьми, представляет собой важный шаг вперёд. Такая программа может предоставить методистам удобные и эффективные инструменты для адаптации учебного материала, коммуникации с детьми и оценки их прогресса. Это поможет создать более инклюзивную и эффективную образовательную среду для слабослышащих детей.

Актуальность данной работы заключается в том, что в настоящее время существует недостаток специализированных компьютерных программ, предназначенных для методистов, работающих со слабослышащими детьми. Исследования в этой области имеют ограниченный объем, и разработка новых инструментов и программ может принести значительную пользу образовательным учреждениям и специалистам.

Цели моей курсовой работы:

* Изучение существующих методов и подходов к обучению слабослышащих детей
* Анализ потребностей методистов и слабослышащих детей
* Разработка компьютерной программы для методистов
* Оценка эффективности программы

# . Анализ потребностей методистов

## Глава 1.1 Выявление требований к программе

Функциональные требования программы:

1. Пользователь должен иметь возможность создать учетную запись пациента, где должны храниться имя, фамилия, а также возрастная группа пациента (4 - 6 лет или 7 - 12 лет).

* В ходе работы медиков было выявлено, что для работы со слабослышащий ребенком, пол не имеет значения, а вот у детей, в вышеупомянутых возрастных категориях, эталонные «здоровые» показатели разные.

2. Пользователь должен иметь возможность хранить в учётной записи пациента записи его речи/отдельных звуков в формате .wav

* Было решено, что необходим выбор конкретного участка записи, для чего при добавлении wav - файла демонстрируется его аудиограмма, где методист может выбрать и автоматически вырезать нужный ему участок.

3. Программа должна хранить информацию о пациенте и записи его речи локально

4. Пользователь должен иметь возможность удалить учетную запись пациента, либо отдельную запись речи

5. После добавления записи речи программа должна рассчитать определенные метрики голоса

6. Пользователь должен иметь возможность при открытии записи речи увидеть рассчитанные характеристики голоса пациента

7. Пользователь должен иметь возможность увидеть не только численные характеристики голоса пациента, но также его какое-то «интересное» представление относительно эталонных данных в возрастной категории пациента для демонстрации его результатов в понятном для пациента виде.

## .2 Ключевые метрики для анализа голоса

1. Частота основного тона: ЧОТ, Fundamental Frequency или F0 – это частота колебания голосовых связок при произнесении тоновых звуков (voiced). При произнесении нетоновых звуков (unvoiced), например говорении шепотом или произнесении шипящих и свистящих звуков, связки не колеблются, а значит эта характеристика для них не релевантна.

2. Среднеквадратическое отклонение частот основного тона: Показатель, отображающий степень разброса частот основного тона периодов относительно среднего значения ЧОТ.

3. Джиттер: Это мера вариативности периода основного тона. Поскольку джиттер оценивает краткосрочную вариацию, она не может быть отнесена к контролируемым (произвольным) изменениям частоты основного тона (ЧОТ). Таким образом, джиттер является показателем стабильности работы фонаторной системы.

5. Шиммер: Это мера, характеризующая степень вариативности амплитуды акустических колебаний в процессе фонации.

6. ВМФ: Длительность (продолжительность) звучания – измеряется в секундах. В среднем максимальное время фонации гласных звуков на средних тонах составляет у детей 10-12 с.

7. Сила звука (громкость): Обусловливаемая величиной амплитуды (размаха) колебаний звучащего тела. Чем больше амплитуда колебательных движений, тем сильнее звучит голос. Сила голоса находится в прямой зависимости от подскладочного давления воздуха, выдыхаемого из легких. Измеряется в децибелах. При обычном разговоре она составляет 40 дБ. В закрытом помещении голос оратора должен достигать не менее 55 дБ, на открытом воздухе – 80 дБ.

8. Форманты: Важнейший параметр, характеризующий спектр (распределение энергии или амплитуды по частотам) речевого сигнала. Их определяют как концентрацию энергию в ограниченной частотной области. Форманта характеризуется частотой, шириной и амплитудой. За частоту форманты принимают частоту максимальной амплитуды в пределах форманты. Другими словами, форманта – это некоторый амплитудный всплеск на графике спектра, а его частота – частота пика этого всплеска.

Для вычисления данных метрик в разрабатываемой мной программе использовался Метод сегментации путём подгонки формы сигнала[[1]](#kirill), описанный и разработанный студентом нашей специальности в ходе его преддипломной практики.

## Глава 1.3 Используемые библиотеки и программные инструменты

Для реализации компьютерной программы в помощь методисту для занятий со слабослышащими детьми, в рамках моей курсовой работы, использовались:

* В качестве языка программирования для разработки данной программы был выбран **Python**[[2]](#python) - это высокоуровневый, интерпретируемый язык программирования, который был разработан в конце 1980-х годов Гвидо ван Россумом. Он отличается простым и понятным синтаксисом, что делает его одним из наиболее доступных языков для изучения и использования.
* **PyCharm** - это интегрированная среда разработки (IDE) для языка программирования Python, разработанная компанией JetBrains. Она предоставляет разработчикам удобное окружение для создания, отладки и развертывания приложений на языке Python.
* **SQLite** (Structured Query Language Lite) - это легковесная, серверная и автономная система управления реляционными базами данных (СУБД). SQLite доступна для различных платформ, включая Windows, macOS, Linux и многие другие. Данные пациента требуется хранить локально, не на сервере, в связи с чем, выбор и пал именно на SQLite
* Для взаимодействия с базой данных из среды разработки использовался **Peewee** - это простая и легковесная ORM (Object-Relational Mapping) библиотека для Python, которая позволяет разработчикам взаимодействовать с базами данных с использованием объектно-ориентированной парадигмы. Она предоставляет удобные инструменты для создания, запроса и обновления данных в базе данных с минимальными усилиями.
* Для построения аудиограммы wav – файла мною была применена библиотека **Librosa** для анализа и обработки аудио с помощью языка программирования Python. Она предоставляет удобные инструменты для извлечения признаков из аудио, визуализации звуковых данных и выполнения различных задач обработки звука.
* Для отображение аудиограммы перед обрезкой wav – файла применялся **Matplotlib.pyplot** - это модуль библиотеки Matplotlib для создания графиков и визуализации данных в языке программирования Python. Он предоставляет широкий спектр возможностей для создания различных типов графиков, диаграмм и иллюстраций.
* Для обрезки аудио использовалась **PyDub** - это библиотека для обработки аудиофайлов в языке программирования Python. Она предоставляет простой и удобный интерфейс для чтения, записи, редактирования и воспроизведения аудиофайлов различных форматов.
* Для создания и отображения пользовательского интерфейса мною использовались **ttkbootstrap** - это модуль библиотеки tkinter, который предоставляет набор стилей оформления визуальных элементов пользовательского интерфейса (UI) для создания приложений с использованием ttk (Themed Tkinter) в языке программирования Python и **customtkinter**[[3]](#custom) - это сторонняя библиотека для создания пользовательского интерфейса (UI) с использованием стандартной библиотеки tkinter в языке программирования Python.

# . Создание программы в помощь методисту для занятий со слабослыщими детьми

## Глава 2. 1 Описание требуемого пользовательского интерфейса

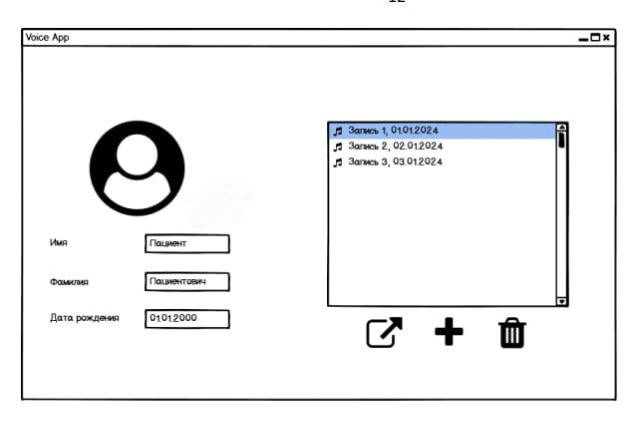
В ходе работы над преддипломным проектом студентом нашей кафедры были выполнены мокапы[[4]](#lesha) для разработки данной программы. На них я и ориентировалась при написании кода для пользовательского интерфейса.

  
Изображение 1. Мокап главной страницы

Главный экран программы имеет вид списка, существующих в базе пациентов.

Здесь пользователь имеет возможность создать нового, удалить существующего пациента или же просто выбрать одного из пациентов из предложенного списка, чтобы посмотреть список его записей.

Изображение 2. Демонстрация карточки пациента



Предположим, пользователя интересует карточка одного из пациентов. Здесь он увидит список записей данного пациента и его личные данные. Также здесь можно добавить новую или удалить существующую запись голоса. Еще из этого экрана, при выборе одной из записей, осуществляется переход к просмотру рассчитанных характеристик этой записи.

## Глава 2.2 Работа с базой данных SQLite

Для реализации хранения данных пациентов, записей, возрастных категорий, а также эталонных данных, как упоминалось раннее, был использован SQLite.

Причины выбора SQLite:

1. Встроенная база данных: SQLite разработана для встраивания в приложения, что означает, что движок базы данных и использующее его приложение тесно интегрированы в одно целое. В отличие от традиционных СУБД, SQLite не работает в режиме клиент-сервер, а представляет собой библиотеку, которую можно включить непосредственно в приложение. Это делает SQLite простым в использовании и не требует отдельного сервера для работы.

2. Легковесный и быстрый: SQLite отличается небольшим размером и низкими требованиями к ресурсам. База данных хранится в одном файле, что делает ее переносимой и удобной для распространения. SQLite также обладает высокой производительностью и быстрым выполнением запросов, особенно при работе с небольшими и средними объемами данных.

3. Поддержка SQL: SQLite полностью совместима со стандартом SQL и поддерживает большинство его функций и команд. Разработчики могут использовать SQL-запросы для создания, изменения и извлечения данных из базы данных SQLite.

4. Надежность и целостность данных: SQLite обеспечивает надежность и целостность данных через поддержку транзакций и механизмов блокировки. Он гарантирует, что изменения в базе данных выполняются атомарно, согласованно, изолированно и долговечно (ACID-свойства).

В ходе работы были созданы следующие таблицы:

1. person – таблица, хранящая основные данные о пациента

1) id: INTEGER Not null Autoincrement Primary Key – уникальный идентификатор пациента. Это же поле является первичным ключом таблицы.

2) name: TEXT Not null – поле, отвечающее за имя пациента

3) surname: TEXT Not null – поле, отвечающее за фамилию пациента

4) category\_id: INTEGER Not null Foreign Key(“category\_id”) references “age\_category(“id”)” – внешний ключ на возрастную категорию пациента, ссылающийся на таблицу, где хранятся данные о категориях.

2. age\_category – таблица, хранящая информацию о том, с какого возраста по какой человек относится к той или иной категории

1) id: INTEGER Not null Autoincrement Primary Key – уникальный идентификатор возрастной категории. Это же поле является первичным ключом таблицы.

2) from\_age: INTEGER Not null – поле, отвечающее за верхний возрастной предел категории

3) to\_age: INTEGER Not null – поле, отвечающее за нижний возрастной предел категории

3. record – таблица, хранящая информацию о записи голоса пациента

1) id: INTEGER Not null Autoincrement Primary Key – уникальный идентификатор пациента. Это же поле является первичным ключом таблицы.

2) person\_id: INTEGER Not null Foreign Key(“person\_id”) references “person(“id”)” – внешний ключ на пациента, ссылающийся на таблицу person, где хранятся данные о пациенте.

3) create\_date: DATE Not null – поле, хранящее информацию о дате создания записи.

4) record: BLOB Not null – поле, хранящее запись голоса пациента в BLOB - формате.

5) info: TEXT Not null – поле, хранящее дополнительную информацию о записи. Содержимое выбирается пользователем.

6) F0: NUMERICAL Not null – поле, хранящее информацию о частоте основного тона.

7) std\_F: NUMERICAL Not null – поле, хранящее информацию о среднеквадратическом отклонении частот основного тона.

8) Jloc: NUMERICAL Not null – поле, хранящее информацию о мере вариативности периода основного тона (джиттер)

9) Sloc: NUMERICAL Not null – поле, хранящее информацию о мере, характеризующей степень вариативности амплитуды акустических колебаний в процессе фонации(шиммера).

4. reference\_values – таблица, содержащая эталонные данные показателей по возрастным категориям.

1) category\_id: INTEGER Not null Foreign Key(“category\_id”) references “age\_category(“id”)” – внешний ключ на возрастную категорию пациента, ссылающийся на таблицу, где хранятся данные о категориях.

2) F0: NUMERICAL Not null – поле, хранящее информацию о эталонной частоте основного тона.

3) std\_F: NUMERICAL Not null – поле, хранящее информацию о эталонном среднеквадратическом отклонении частот основного тона.

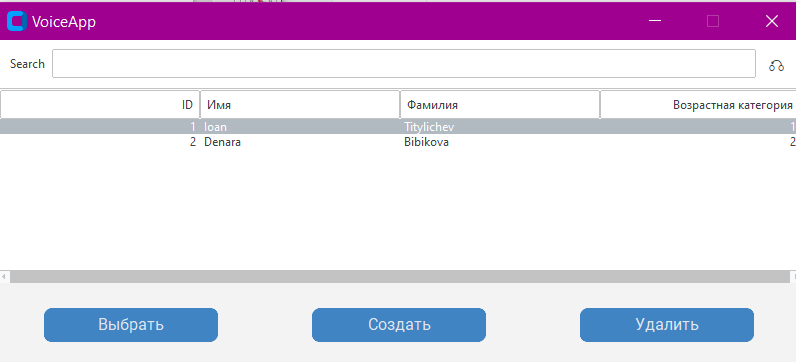
4) Jloc: NUMERICAL Not null – поле, хранящее информацию о эталонной мере вариативности периода основного тона (джиттер)

5) Sloc: NUMERICAL Not null – поле, хранящее информацию о эталонной мере, характеризующей степень вариативности амплитуды акустических колебаний в процессе фонации(шиммера).

## Глава 2.3 Разработка программы

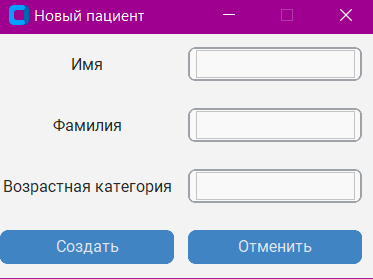
Основным классом моей программы является App class, наследуемый от customtkinter.Ctk. Данный класс предназначен для создания и управления основным окном приложения для работы с пациентами. Он предоставляет функциональность для отображения данных, создания, открытия и удаления записей пациентов.

* \_\_init\_\_(self, \*args, \*\*kwargs)
  + Параметры:
    - \*args, \*\*kwargs: Дополнительные аргументы и ключевые аргументы, передаваемые в конструктор родительского класса ctk.CTk.
  + Действие:
    - Инициализирует объект класса App.
    - Создает переменные для хранения ссылок на кнопки, таблицу и всплывающее окно.
    - Вызывает методы config(), init(), pack() для конфигурации приложения и создания виджетов.
* update\_table(self)
  + Возвращает:
    - Список списков, содержащий данные о пациентах из базы данных.
  + Действие:
    - Запрашивает и извлекает данные о пациентах из базы данных.
    - Преобразует данные в формат, подходящий для отображения в таблице.
* create\_callbck(self)
  + Действие:
    - Проверяет, существует ли всплывающее окно или было ли оно уничтожено.
    - Если окно не существует или было уничтожено, создает новое всплывающее окно Popup.
    - В противном случае фокусируется на существующем всплывающем окне.
* open\_callbck(self)
  + Действие:
    - Проверяет, существует ли всплывающее окно или было ли оно уничтожено.
    - Если окно не существует или было уничтожено, получает выбранную строку в таблице, извлекает идентификатор пациента и создает новое всплывающее окно RecordsList для отображения записей пациента.
    - В противном случае фокусируется на существующем всплывающем окне.
* delete\_callbck(self)
  + Действие:
    - Получает выбранную строку в таблице.
    - Извлекает идентификатор пациента из выбранной строки.
    - Создает новое всплывающее окно Delete для удаления пациента.
* config(self)
  + Действие:
    - Конфигурирует приложение, устанавливая его позицию, заголовок и возможность изменения размеров окна.
* pack(self)
  + Действие:
    - Размещает таблицу и кнопки в основном окне приложения.

  
Изображение 3. Главный экран программы

Следующий класс Popup является наследником класса ctk.CtkTopLevel и представляет собой всплывающее окно для создания нового пациента. Описание его методов:

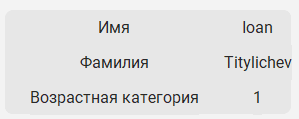
* \_\_init\_\_(self, parent, \*args, \*\*kwargs)
  + Параметры:
    - parent: Родительское окно, из которого вызывается всплывающее окно.
    - \*args, \*\*kwargs: Дополнительные аргументы и ключевые аргументы, передаваемые в конструктор родительского класса ctk.CTkToplevel.
  + Действие:
    - Инициализирует объект класса Popup.
    - Устанавливает позицию и заголовок окна.
    - Создает переменные StringVar для хранения значений имени, фамилии и возрастной категории.
    - Вызывает метод init() для создания и размещения виджетов в окне.
    - Устанавливает нерасширяемые размеры окна.
    - Упаковывает виджеты в окне с использованием метода pack().
* init(self)
  + Действие:
    - Инициализирует и создает виджеты (Entry, Label, Button) для ввода данных о новом пациенте.
    - Связывает виджеты с соответствующими переменными StringVar.
    - Задает текст и команды для кнопок.
* pack(self)
  + Действие:
    - Устанавливает расположение виджетов в окне с использованием сетки.
    - Устанавливает метки и поля ввода в нужные ячейки сетки.
    - Устанавливает кнопки в нужные ячейки сетки.
* save\_user(self)
  + Действие:
    - Создает объект Person с данными из полей ввода.
    - Вызывает метод save() объекта Person для сохранения пациента.
    - Вставляет новую строку с данными пациента в таблицу родительского окна.
    - Обновляет родительское окно.
    - Закрывает всплывающее окно.
* cancel(self)
  + Действие:
    - Закрывает всплывающее окно без сохранения данных.

  
Изображение 4. Экран создания нового пациента

Следующий класс ProfileFrame является наследником класса ctk.CtkFrame и представляет собой виджет для отображения профиля пациента.

Его единственный метод:

* \_\_init\_\_(self, master, \*\*kwargs)
  + Параметры:
    - master: Родительский виджет, в котором будет размещен ProfileFrame.
    - \*\*kwargs: Дополнительные ключевые аргументы, передаваемые в конструктор родительского класса ctk.CTkFrame.
  + Действие:
    - Инициализирует объект класса ProfileFrame.
    - Создает виджеты ctk.CtkLabel для отображения имени, фамилии и возрастной категории пациента.
    - Устанавливает текст для меток имени, фамилии и возрастной категории, используя информацию из родительского виджета master.
    - Размещает метки на сетке внутри ProfileFrame.

  
Изображение 5. Отображение личной информации в карточке пациента

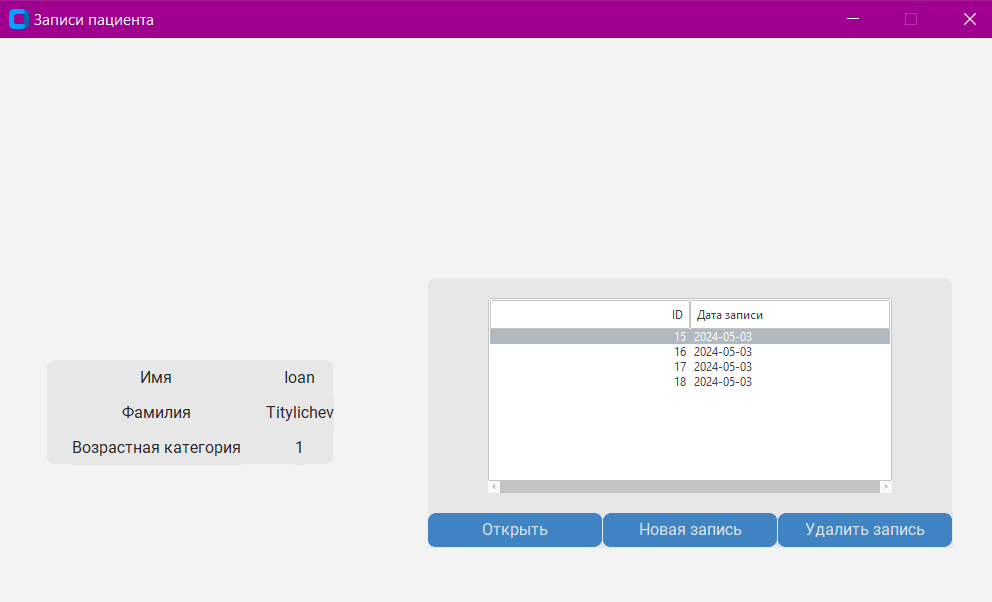
Класс Delete является наследником класса ctk.CtkToplevel. Класс Delete предназначен для создания всплывающего окна и удаления пациента из базы данных.

* \_\_init\_\_(self, parent, pt\_id, \*args, \*\*kwargs)
  + Параметры:
    - parent: Родительский виджет, к которому принадлежит всплывающее окно.
    - pt\_id: Идентификатор пациента, который должен быть удален.
    - \*args, \*\*kwargs: Дополнительные аргументы и ключевые аргументы, передаваемые в конструктор родительского класса ctk.CTkToplevel.
  + Действие:
    - Инициализирует объект класса Delete.
    - Сохраняет идентификатор пациента и ссылку на родительский виджет.
    - Вызывает метод delete\_user() для удаления пациента.
* delete\_user(self)
  + Действие:
    - Получает объект пациента из базы данных на основе идентификатора пациента.
    - Удаляет пациента из базы данных по его идентификатору.
    - Удаляет строку с пациентом из таблицы в родительском виджете.
    - Обновляет родительский виджет после удаления пациента.
    - Уничтожает всплывающее окно.

Класс RecordFrame является наследником класса ctk.CtkFrame и представляет собой виджет для отображения записей пациента.

Методы:

* \_\_init\_\_(self, master, row\_data, \*\*kwargs)
  + Параметры:
    - master: Родительский виджет, в котором будет размещен RecordFrame.
    - row\_data: Данные для заполнения таблицы записей.
    - \*\*kwargs: Дополнительные ключевые аргументы, передаваемые в конструктор родительского класса ctk.CTkFrame.
  + Действие:
    - Инициализирует объект класса RecordFrame.
    - Создает таблицу Tableview для отображения записей пациента на основе переданных данных row\_data.
    - Создает кнопки для открытия, добавления и удаления записей.
    - Размещает таблицу и кнопки на сетке внутри RecordFrame.
* trim\_audio(self)
  + Действие:
    - Получает индексы начала и конца выбранных точек.
    - Создает имя файла для усеченной записи на основе идентификатора пациента и номера записи.
    - Загружает аудиофайл и усекает его в соответствии с выбранными точками.
    - Экспортирует усеченное аудио в файл формата wav.
    - Возвращает усеченное аудио.
* add\_record(self)
  + Действие:
    - Запрашивает у пользователя выбор аудиофайла.
    - Загружает аудиофайл и выводит информацию о его длительности.
    - Отображает аудиограмму аудиофайла и позволяет пользователю выбрать точки для усечения.
    - Вызывает метод trim\_audio() для усечения аудио в соответствии с выбранными точками.
    - Если усеченный файл существует, открывает его, читает его содержимое и создает объект записи Record.
    - Сохраняет объект записи в базе данных.
    - Вызывает методы для обработки изображения и расчета параметров записи.
* delete\_record(self)
  + Действие:
    - Получает выбранную запись из таблицы.
    - Получает идентификатор записи.
    - Создает экземпляр класса DeleteRecord для удаления записи.
* open\_record(self)
  + Действие:
    - Получает выбранную запись из таблицы.
    - Получает идентификатор записи.
    - Создает экземпляр класса ShowRecord для отображения записи.

  
Изображение 6. Экран карточки пациента

Класс DeleteRecord предназначен для удаления записи из базы данных.

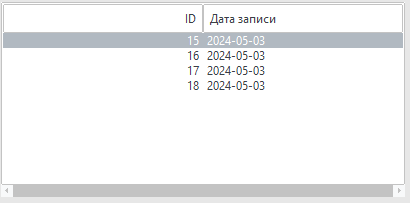
Методы:

* \_\_init\_\_(self, rec\_id)
  + Параметры:
    - rec\_id: Идентификатор записи, которую следует удалить.
  + Действие:
    - Инициализирует объект класса DeleteRecord.
    - Сохраняет идентификатор записи.
    - Вызывает метод delete\_record() для удаления записи.
* delete\_record(self)
  + Действие:
    - Получает объект записи из базы данных на основе идентификатора записи.
    - Удаляет запись из базы данных по её идентификатору.

Класс RecordsList представляет собой верхний уровень (top-level) виджет, отображающий список записей пациента.

Методы:

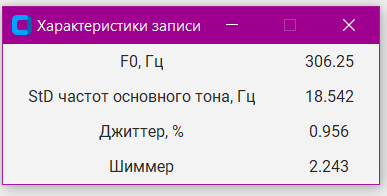
* \_\_init\_\_(self, parent, patient\_id, \*args, \*\*kwargs)
  + Параметры:
    - parent: Родительский виджет, в котором будет размещен RecordsList.
    - patient\_id: Идентификатор пациента, для которого отображаются записи.
    - \*args, \*\*kwargs: Дополнительные аргументы, передаваемые в конструктор родительского класса ctk.CTkToplevel.
  + Действие:
    - Инициализирует объект класса RecordsList.
    - Сохраняет родительский виджет и конфигурирует виджет.
    - Получает объект пациента на основе переданного идентификатора.
    - Получает данные записей пациента и сохраняет их в rowdata.
    - Конфигурирует сетку виджета, задавая веса для столбцов и строк.
    - Создает объект ProfileFrame для отображения профиля пациента и размещает его на сетке.
    - Создает объект RecordFrame для отображения списка записей пациента и размещает его на сетке.
* config(self)
  + Действие:
    - Конфигурирует виджет RecordsList.
    - Задает размеры виджета, делая его нерасширяемым по горизонтали и вертикали.
    - Устанавливает заголовок окна.

  
Изображение 7. Демонстрация списка записей пациента

Класс ShowRecord представляет собой верхний уровень (top-level) виджет, отображающий характеристики записи.

Методы:

* \_\_init\_\_(self, parent, record\_id, \*args, \*\*kwargs)
  + Параметры:
    - parent: Родительский виджет, в котором будет размещен ShowRecord.
    - record\_id: Идентификатор записи, характеристики которой отображаются.
    - \*args, \*\*kwargs: Дополнительные аргументы, передаваемые в конструктор родительского класса ctk.CTkToplevel.
  + Действие:
    - Инициализирует объект класса ShowRecord.
    - Инициализирует переменные person\_id и record\_id.
    - Сохраняет родительский виджет и конфигурирует виджет.
    - Вызывает метод init() для инициализации характеристик записи.
    - Вызывает метод pack() для расположения характеристик записи.
    - Вызывает метод show\_image() для отображения изображения, связанного с записью.
* init(self)
  + Действие:
    - Получает объект записи из базы данных на основе идентификатора записи.
    - Сохраняет идентификатор пациента, связанного с записью.
    - Создает объекты ctk.CtkLabel для отображения характеристик записи.
* pack(self)
  + Действие:
    - Располагает созданные объекты ctk.CtkLabel на виджете ShowRecord, задавая их положение на сетке.
* show\_image(self)
  + Действие:
    - Открывает изображение, связанное с записью, используя путь к файлу.
    - Отображает изображение.
* config(self)
  + Действие:
    - Конфигурирует виджет ShowRecord.
    - Запрещает изменение размеров виджета.
    - Устанавливает заголовок окна.

  
Изображение 8. Демонстрация характеристик записи

Класс ImageWork предоставляет методы для работы с изображением, которое будет показываться пациенту при его новой записи.

Методы:

* \_\_init\_\_(self, image\_path)
  + Параметры:
    - image\_path: Путь к изображению.
  + Действие:
    - Инициализирует объект класса ImageWork.
    - Сохраняет путь к изображению в переменную image\_path.
* adjust\_saturation(self, factor, person\_id, record\_id)
  + Параметры:
    - factor: Фактор насыщенности, определяющий величину изменения насыщенности изображения.
    - person\_id: Идентификатор пациента.
    - record\_id: Идентификатор записи.
  + Действие:
    - Открывает изображение с использованием сохраненного пути image\_path.
    - Создает объект изменения изображения enhancer с помощью класса ImageEnhance.Color.
    - Изменяет насыщенность изображения с помощью метода enhance() и сохраняет результат в enhanced\_image.
    - Отображает измененное изображение.
    - Сохраняет измененное изображение по пути "resources/progress\_image\_" + str(person\_id) + "\_" + str(record\_id) + ".png".
* count\_progress(self, person\_id)
  + Параметры:
    - person\_id: Идентификатор пациента.
  + Действие:
    - Получает объект пациента из базы данных на основе идентификатора.
    - Получает идентификатор категории пациента.
    - Получает список записей пациента и список эталонных значений.
    - Выделяет последнюю запись пациента rec1 и первое эталонное значение rec2.
    - Вычисляет разницу между характеристиками F0, std\_F, Jloc и Sloc между rec1 и rec2.
    - Вычисляет евклидово расстояние между разницами характеристик.
    - Вызывает метод adjust\_saturation() для настройки насыщенности изображения на основе расстояния и сохраняет результат.

# Заключения

# Список использованной литературы

1.

2. Официальный сайт языка программирования Python <https://www.python.org/>

3. Документация customtkinter <https://pypi.org/project/customtkinter/0.3/>

4.

# Приложения