**Архитектура проекта.**

Проект состоит из двух компонентов, связанных между собой таблицей.

1) scanner.py -- обрабатывает нужные файлы python, раскладывает их на функции, классы и глобальные переменные.

2) gui.py -- осуществляет поиск и вывод файлов с необходимыми сущностями.

Остальные файлы:

* scan\_one.py содержит функцию, генерирующую скомпелированные файлы, данные с которых scanner.py заносит в таблицу.
* search.py осуществляет поиск по таблице по параметрам, заданным пользователем.
* UI файлы – дизайн GUI
* sources.txt – файл, содержащий пути папок, среди файлов которых нужно осуществлять поиск.

**Актуальность**

Промышленные программисты много работают с чужим кодом. Они используют чужие проекты, а свои делают с командой, и поэтому бывает тяжело найти какой-нибудь элемент...

**Цель**

**Задачи**

**Структура проекта**

**База данных**

Я решил использовать распространённый и лёгкий (по сравнению с другими) инструмент организации и использования баз данных -- SQLite3. Работа с ним осуществляется на языке SQL. Я изучил SQL и модуль sqlite3 для отпраки запросов прямо в программе.

**Сканер**

Scanner создаёт и готовит базу данных, потом обрабатывает данные файлы и заносит информацию в таблицы.

Вначале создаётся база данных и запускается функция create\_tables, которая создаёт четыре таблицы в базе данных.

* таблица "files" для списка просканированных файлов
* таблица "global\_vars(fname, name, line\_no)" для списка глобальных переменных
* таблица "classes(fname, name, doc\_string, line\_no)" для списка классов
* таблица "functions(fname, name, class\_name, doc\_string, line\_no)" для списка функций

В последних трёх таблицах поле "fname" -- внешний ключ к таблице "files"

Пути к файлам python находятся в отдельном txt файле. Я использовал библиотеку pathlib для того, чтобы найти все файлы python -- если дан путь вкладки, то программа открывает эту папку и все вложенные папки, чтобы найти все вложенные файлы python.

Теперь эти файлы нужно обработать -- для этого я использую стандартный модуль ast. Он разбирает программу в представление AST -- Abstract Syntax Tree, Абстрактное синтаксическое дерево. Программа python делится на верхние сущности и её подчинённые -- например верхняя сущность class имеет наследуемый класс, классовые переменные, методы, которые, в свою очередь, имеют аргументы, тело функции, в котором содержатся локальные переменные... И по такому принципу выстраивается дерево, по которому моя программа совершает рекурсивный обход для сбора данных.

Я написал вспомогательную программу scan\_one, главная функция-генератор которой scan\_py\_file получает в качестве аргумента путь к файлу, который нужно обработать. Функция-генератор -- это такая сущность в python, которая представляет собой коллекцию, заполняющуюся в реальном времени.

Все данные собраны и занесены в таблицу, теперь нужно создать поиск.

**GUI**

Для диалога с пользователем я использовал фреймворк PyQt5. Этот модуль работает с ui файлами -- файлами деклоративной разметки, в которых содержится информация о расположении объектов и их параметров в приложении.

Класс основного окна MainWindow, унаследованный от QMainWindow, представляет главное окно, в котором пользователь вводит параметры поиска в окне в объект QLineEdit, а тип сущности, которую ищет пользователь, задаётся с помощью QRadioButton. Затем пользователь нажимает кнопку и открывается окно с таблицей, в которой содержится строчка и путь к файлу, сущность в котором соответствует параметрам поиска. Если нажать на путь файла, откроется окно с программой на строчке с искаемой сущностью.