# ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ РАЗРАБОТКИ ПРОГРАММНО-АЛГОРИТМИЧЕСКОГО КОМПЛЕКСА АНАЛИЗА ОКТ-ИЗОБРАЖЕНИЙ СЕТЧАТКИ И ЭКСПЕРИМЕНТАЛЬНАЯ АПРОБАЦИЯ РАЗРАБОТКИ

## Выбор инструментов разработки

Задачу разработки программно-алгоритмического комплекса анализа ОКТ-изображений сетчатки возможно решить, используя различные наборы технологий. Однако, вдумчивый и обоснованные выбор инструментария способствует упрощению и ускорению разработки программно-алгоритмического комплекса.

### Обоснования выбора языка программирования

Основным языком программирования, использующимся для разработки систем обработки изображения, был выбран язык Python. Язык Python обладает лаконичным и легко читаемым синтаксисом, что позволяет легче вносить изменения в код проекта на всех этапах разработки. Также, Python обладает обширным набором специализированных библиотек, упрощающих обработки изображений и работу c проектами, использующими машинное обучение.

Также, использование языка Python позволяет обеспечить кроссплатформенность разрабатываемого программно-алгоритмического комплекса, так как Python работает на платформах Windows, Linux и macOS, суммарно охватывающих наибольшее количество пользователей ЭВМ.

### Выбор библиотек для обработки изображений

На данной этапе разработки программно-алгоритмического комплекса было решено использовать такие библиотеки, как:

* OpenCV – библиотека компьютерного зрения, содержащая более 2500 алгоритмов для работы и обработки изображений. Использование данной библиотеки позволяет упросить процесс разработки модулей предобработки, позволяя использовать уже готовые алгоритмы обработки изображений.
* Ultralytics – библиотека компьютерного зрения на основе искусственного интеллекта. Использование данной библиотеки позволяет упростить процесс обучения и использования моделей архитектуры YOLO.
* MatPlotLib – библиотека визуализации данных в виде статических графиков. Данная библиотека позволяет упросить визуализацию результатов работы каждого блока модуля, что требуется для контроля качества получаемых результатов обработки.

### Выбор инструментов разработки пользовательского интерфейса

В целях обеспечения кроссплатформенного использования разрабатываемого программно-алгоритмического комплекса пользовательский интерфейс было решено разработать в виде вэб-приложения. Данный подход позволяет использовать разрабатываемый программно-алгоритмический комплекс на любой платформе, имеющий какой-либо браузер.

Для разработки пользовательского интерфейса был использован язык TypeScript. TypeScript является расширенной версией JavaScript. TypeScript позволяет повысить надёжность, читаемость и поддерживаемость кода, снизить вероятность ошибок и облегчить масштабирование проекта.

### Выбор инструментов для разработки серверной инфраструктуры

Пользовательский интерфейс предоставляет визуальные инструменты взаимодействия с разрабатываемым программно-алгоритмическим комплексом. Вычислительная часть разрабатываемого комплекса расползается в отдельном логическом блок – сервере.

Для разработки серверной инфраструктуры была выбрана библиотека Flask. Flask является минималистичным веб-фреймворком. Он предоставляет простую и гибкую структуру, позволяющую быстро разработать как прототип, так и полнофункциональное веб-приложение.

## Разработки модуля предобработки ОКТ-изображений

В листинге 1 представлен пример программы, выполняющей предобработку обрабатываемого ОКТ-изображения.

Листинг 1 – Программа модуля предобработки

|  |
| --- |
| image = cv2.imread(f"{dir}/{file}")      converted\_image = prep.convert\_to\_gray(image)      filtered\_image = prep.smooth\_image(converted\_image)      sobel\_image = prep.sobel(filtered\_image)      ret2, binary = prep.threshold(sobel\_image)      element = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))      binary = prep.morph\_open(binary, element)      binary = prep.morph\_close(binary, element)      contours = prep.get\_contours(binary)      resized = prep.resize\_image(image, contours) |

Далее, каждый из блоков, составляющих разрабатываемый модуль, рассмотрены отдельно.

### Реализация преобразования к чёрно-белому изображению

В листинге 2 представлен пример кода, преобразующего обрабатываемого ОКТ-изображение к чёрно-белому.

Листинг 2 – Преобразование ОКТ-изображения к чёрно-белому

|  |
| --- |
| def convert\_to\_gray(self, image):          return cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY) |