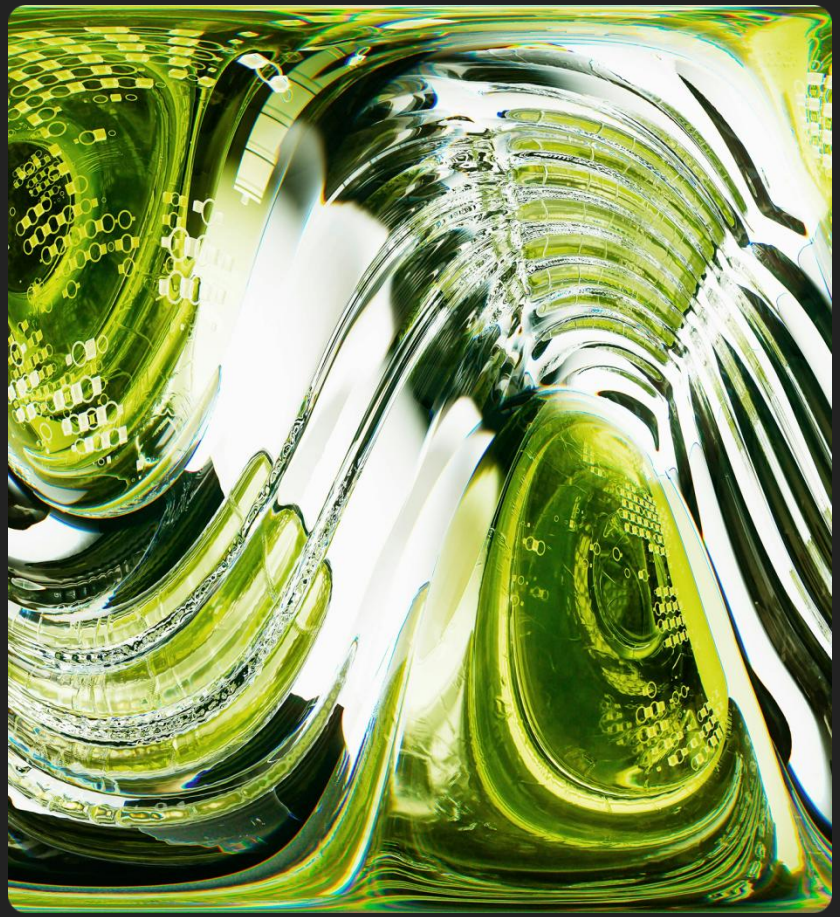
Хакатон 3.0

Сириус
Научно-технологический университет



AIm

Университет Сириус

Распределение работы в команде

Дарья Шайнова
- PM

Михаил Коломиец
- Front-end

Роман Батанин
- ML

Энкира Ходжгорова
- UI/UX design

Артем Григорьев
- Back-end

О проекте

Задача: Разработать приложение, которое на основе входных данных (карточек контейнеров) распределяет контейнеры по поездам с учётом заданных критериев оптимальности.

Реализация проекта: В приложении есть функция, которая позволяет задать собственные параметры оптимальности, и выбрать во сколько раз приоритетнее тот или иной критерий (от 1 до 5).

Анализ целевой аудитории и её потребностей

Компании, занимающиеся логистикой и грузоперевозкой.

Портрет пользователя

Параметр	Характеристики
Должность	Логисты, операторы распределения, диспетчеры на ЖД-терминалах, руководители логистических отделов
Что им важно	Надёжность, понятность, контроль приоритета и веса, экономия времени и исключение “человеческого фактора”
Ожидания	Простое, предсказуемое решение, которое работает сразу. Без лишних кнопок и без 100-страничных инструкций

Анализ конкурентов / аналогов / используемых технологий

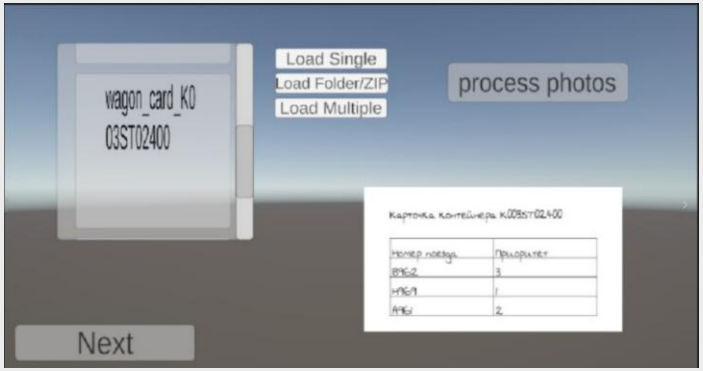
Название продукта	1С:Логистика.Управление перевозками	ИТ-Платформа “Логистика” от Группы Синара	Разработка на Python (индивидуальная)
Описание	Модуль в составе платформы 1С для управления логистическими операциями	Современное решение для цифровизации перевозок (в т.ч. ж/д)	Скрипты и ПО под конкретные нужды
Преимущества	Интеграция с 1С, гибкие настройки маршрутов	Интеграция с ЖД-инфраструктурой и крупными логистическими цепочками.	Гибкость, скорость разработки, можно внедрить любые алгоритмы (ИИ, оптимизацию)
Недостатки	Сложный интерфейс, нет адаптации под контейнеры на ж/д, дорогая интеграция	Интеграция только на уровне предприятий, высокая стоимость	Требует разработки интерфейса, поддержки и тестирования

Входные данные

Цель алгоритма:
Алгоритм нужен, чтобы автоматически распределить контейнеры по поездам так, чтобы приоритетные грузы не задерживались, а нагрузка между поездами была максимально сбалансированной

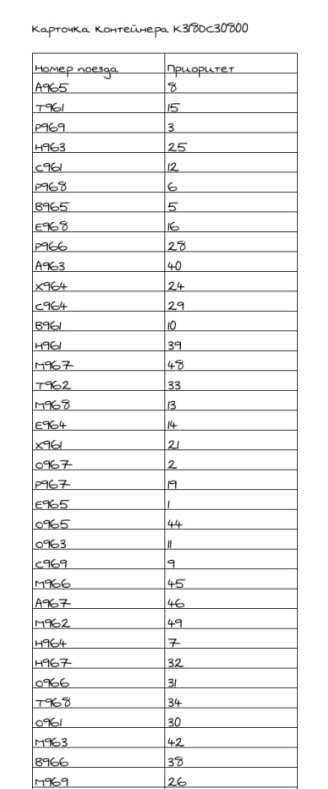
На вход скрипту подаются данные из таблицы: номер контейнера, номер поезда, вес и приоритет.
Пример входных данных:

Интерфейс приложения



Интерфейс приложения

Преобразование картинки в текст



Карточка Контейнера K3B0C30B00

Номер поезда	Приоритет
A965	8
T961	15
P969	3
H963	25
C961	12
P968	6
B965	5
E968	16
P966	28
A963	40
X964	24
C964	29
B961	10
H961	39
M967	48
T962	33
M968	13
E964	14
X961	21
O967	2
P967	19
E965	1
O965	44
O963	11
C969	9
M966	45
A967	46
M962	49
H964	7
H967	32
O966	31
T968	34
O961	30
M963	42
B966	38
M969	26

=== wagon_card_K001RF02400.png ===
Карточка контейнера K001RF02400
Номер поезда Приоритет
A965 41
T961 8
P969 2
H963 18
C961 16
P968 15
B965 9
E968 7
P966 35
A963 6
X964 38
C964 28
B961 3
H961 48
M967 50
T962 14
M968 45
E964 33
X961 37

```
[ ]  def preprocess_image(image_path):
        try:
            # Загрузка изображения
            img = cv2.imread(image_path)
            if img is None:
                raise ValueError(f"Не удалось загрузить изображение по пути: {image_path}")

            # Преобразование в оттенки серого
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            # Увеличение контрастности
            clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
            contrast = clahe.apply(gray)

            # Применение пороговой обработки
            _, thresh = cv2.threshold(contrast, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

            # Удаление шума
            denoised = cv2.fastNlMeansDenoising(thresh)

            return denoised
        except Exception as e:
            print(f"Ошибка при предобработке изображения: {e}")
            return None

    def extract_text(image):
        try:
            # Распознавание текста с помощью Tesseract
            custom_config = r'--oem 3 --psm 6'  # PSM 6 для табличного текста
            text = pytesseract.image_to_string(image, lang='rus+eng', config=custom_config)
            return text
        except Exception as e:
            print(f"Ошибка при извлечении текста: {e}")
            return ""

    def parse_table(text):
```

Алгоритм / логика

```
void rec(int x){
    if (x == kol_Kont && fin()){
        for (int l=0; l<kol_Poezdov-1; l++){
            if (Kont_Poezd_answer[l]!=Kont_Poezd_answer[l+1]){
                return;
            }
        }
        if (delta()  * cof + raz() <= answer_int){
            Kont_Poezd_answer  = Poezd_Kont_prom;
            answer_int = delta() * cof + raz();
        }
    }

    Kont_prom_flag[x] = false;

    for (int i=0; i<kon_poezda[x].size();i++){
        Poezd_Kont_prom[kon_poezda[x][i]].push_back(x);
        rec(x+1);
        Poezd_Kont_prom[kon_poezda[x][i]].pop_back();
    }

    return;
}
```

Результаты

- Распознавание текста
- Анализ и обработка данных
- Алгоритм распределения контейнеров по поездам
- Автоматизация принятия решений
- Гибкость и масштабируемость

Дальнейшее развитие проекта

Наш проект помогает решать реальную проблему — как быстро и правильно распределить контейнеры между поездами, чтобы всё работало без сбоев. Сегодня в логистике каждый день — это десятки решений, и чем больше объёмы перевозок, тем выше цена ошибки.
По мере того как логистика становится всё более цифровой, такие решения, как наше, будут не просто полезными — они станут необходимостью.

01

Интеграция с базами данных и API

- Загрузка данных контейнеров напрямую из систем управления складом или терминала;
- Автоматический импорт расписаний поездов и свободных мест

02

Масштабирование

- Поддержка распределения на уровне нескольких станций и регионов
- Настройка шаблонов распределения под разные типы грузов

03

Автоматическая генерация документов

- Печать накладных, ведомостей, отчётов по приоритетам
- Экспорт в Excel, PDF, XML по шаблонам предприятий

04

Связь с графиками движения поездов

- Программа может учитывать расписание и перестраивать приоритеты
- Автоматическое предложение альтернативного поезда

Спасибо за внимание!