УДК 004.3

Киричек Г.Г.
Запорізький національний технічний університет

Курай В.І.
Запорізький національний технічний університет

КЛІЄНТ-СЕРВЕРНА СИСТЕМА ВИЗНАЧЕННЯ ОБ'ЄКТІВ

У роботі розглянуто питання розбиття й порівняння зображень з метою реалізації високоефективного доступу до елементів. Для досягнення поставленої мети використовується рекурсивне розбиття двовимірного простору, робота з клієнт-серверною архітектурою та фреймворком Django.

Ключові слова: *api, сегментація, розпізнавання об'єктів, сервер, клієнт.*

Постановка проблеми. Візуальна інформація – тип інформації, що сприймається, інтерпретується й обробляється людським мозком, третина кортикальної ділянки якого присвячена її обробці. Цифрова обробка зображень, як комп'ютерна технологія, здійснює автоматичну обробку, маніпулювання та інтерпретацію візуальної інформації. Технології обробки зображень відіграють важливу роль у багатьох аспектах повсякденного життя, в техніці, науці й таких галузях, як медична діагностика, дистанційне зондування, робототехніка тощо. Тому питання компактного подання зображень і забезпечення високоефективного доступу до елементів під час ідентифікації їх відмінностей є ключовими вимогами до подібних систем.

Аналіз останніх досліджень і публікацій. Технології та інструменти, які застосовують під час проведення досліджень у цьому напрямі, є такими: штучні нейронні мережі, перцептивні хеш-алгоритми, бібліотеки для роботи із зображеннями типу OpenCV та способи подання зображень, що зберігаються, у вигляді структур даних [1; 2]. Ділянка обробки цифрових зображень розвивається за двома напрямами: вдосконалення механізмів обробки й маніпуляції над графічною інформацією та аналіз даних для їх подальшого сприйняття машиною [3].

Процес кластеризації зображень, пошуку в них однорідних ділянок є сегментацією, яка вважається першим етапом аналізу зображень. Наведемо її загальну математичну модель. Нехай $D(m*n)$ – растр або ділянка поля зору, на якій задано зображення. $B(i,j)$; $D_k \subset D$ – k-го об'єкта, де k = 1, 2, …, s; $D_\phi \subset D$ – ділянка фону. Уважаємо, що растр або поле зору можна представити як у формулі 1.

$$D_1 \cup D_2 \cup \ldots \cup D_\phi = D, D_i \cup D_j = \varnothing, i \neq j\,. \qquad (1)$$

Розглянемо дискретне зображення $B(i,j)$, де і = 0, …, m, j = 0, …, n, яке є сукупністю зображень окремих об'єктів і фону та наведене у формулі 2.

$$B(i,j) = H_1(i,j) + \ldots + H_s(i,j) + H_\phi(i,j)\,, \qquad (2)$$

де s – кількість об'єктів; $H_k(i,j)$ – зображення k-го об'єкта, k = 1, 2, …, s; $H_\phi(i,j)$ – зображення фону. Задача сегментації полягає в побудові предикату, наведеного у формулі 3.

$$\pi(i,j) = \begin{cases} k, якщо (i,j) \in D_k \\ 0, якщо (i,j) \in D_\phi \end{cases}. \qquad (3)$$

На етапі сегментації відбувається групування розрізнених ділянок чи фрагментів зображення в ділянку, яка належить одному об'єкту, або поділ ділянки зображення на ділянки, що належать різним об'єктам [3]. Групування здійснюється за різними ознаками, такими як яскравість; колір; текстура; рух в одному напрямку, з однаковою швидкістю тощо. Визначивши методи сегментації, які основані на пошуку меж регіонів; бінаризації зображень; марківському випадковому полю та пошуку регіонів, зупинимося на тих методах, які знаходять регіони безпосередньо, об'єднуючи сусідні пікселі в регіони за схожістю параметрів. Ці методи лежать в основі методів: злиття регіонів; злиття-розщеплення регіонів; «водорозділу». Метод злиття-розщеплення полягає в розщеплені зображення на квадрати й проведенні аналізу їх однорідності. Результат злиття-розщеплення – структура з інформацією про квадрати, наприклад, дерево квадрантів, у якому кожен внутрішній вузол має 4 нащадки. Метод дерева квадрантів – структури даних, які використовуються для подання двомірних просторових даних, це дерево, зі збільшенням глибини якого збільшується кількість елементів дерева. Алгоритм має таку залежність часу виконання O (u + p + q), де u – кількість полігонів, p – периметр полігонів, роздільна здатність зображення [4]. Причини вибору цього методу розбиття такі: компактне подання зображення; високоефективний доступ до елементів. Як програмний метод розглянемо Python – інтерпретовану об'єктна-

орієнтовану мову програмування, що підтримує пакети модулів і декілька парадигм програмування: об'єктна-орієнтовану, процедурну, функціональну та аспектно-орієнтовану [5]. На вибір Python вплинули підтримка Python об'єктна-орієнтованого підходу, простота синтаксису та наявність убудованих функцій і структур даних. Далі проведемо вибір рішень і технологій для реалізації клієнт-серверної архітектури системи.

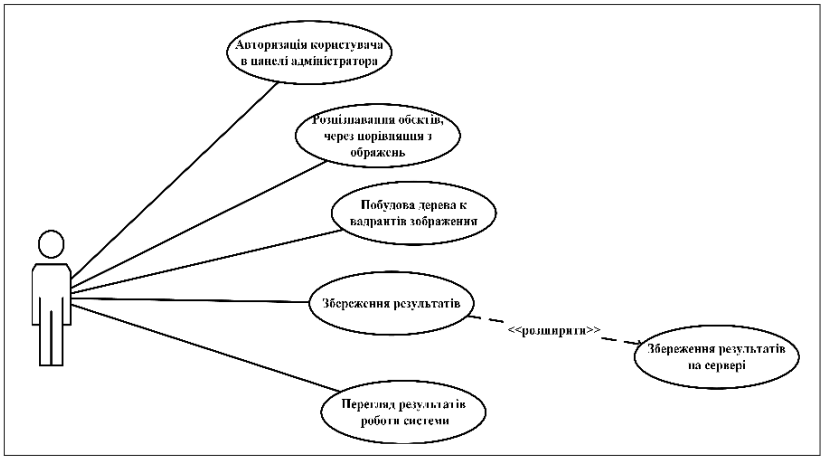

Рис. 1. Діаграма варіантів використання системи

Постановка завдання. Метою роботи є впровадження клієнт-серверної системи визначення об'єктів з використанням методу дерева квадрантів. Об'єктом дослідження є процес реалізації клієнт-серверної системи визначення об'єктів. Предметом – моделі, методи та інструментальні засоби обробки цифрових зображень і реалізації клієнт-серверних систем. Методи дослідження базуються на моделюванні системи та схем взаємодії її модулів, вимірах основних характеристик системи й характеристик дерева квадрантів. Для досягнення поставленої мети визначено завдання: проведення досліджень і реалізація методу обробки зображень; рекурсивне розбиття зображень за допомогою дерева квадрантів; порівняння результатів розбиття двох зображень та отримання результуючого зображення; автоматизація процесу обробки зображень з використанням клієнт-серверної архітектури. Параметри для роботи системи: вхідні зображення, порогові значення кольору й розміру квадрата – повинні передаватися за допомогою використання API запитів.

Виклад основного матеріалу дослідження. Конкретні завдання в проекті реалізуються через обрану методологію, що визначає процес розроблення. Популярними є каскадна, інкрементна та гнучка моделі. До гнучкої зараховують методи: Scrum і Kanban. Scrum розбиває складні завдання та візуалізує їх у робочому процесі. Його краще застосовувати, коли потрібно регулярно подати нові версії та оновлення клієнтам. Тому використаємо Scrum, яка є адаптивною, орієнтована на роботу в невеликих командах і проста в застосуванні.

Створення архітектури є процесом формування структурованого рішення, що відповідає технічним та операційним вимогам. Ознайомившись із архітектурними стилями, прийняли рішення використати два шаблони для реалізації системи: клієнт-серверну архітектуру й об'єктна-орієнтований підхід. Це обґрунтовано вимогами до системи: збереженням результатів роботи на сервері; взаємодією за допомогою API запитів; залученням самостійних, придатних для повторного використання об'єктів під час створення додатків.

Так як мовою програмування обрано Python, у роботі розглянуто два веб-фреймворки: Django та Flask [6]. Django – для веб-додатків мовою Python, який використовує шаблон проектування MVC. Він складається з одного або декількох додатків, що ізолюють один від одного. Для роботи з базою даних (далі – БД) Django використовує власний ORM (Object Relational Mapping), у якому модель даних описується класами Python і по якій генерується схема бази даних. Flask – мікрофреймворк, основою якого є інструментарій Werkzeug і Jinja2. Поширюється відповідно до умов ліцензії BSD. Django дає більш можливостей для швидкого розроблення складних веб-додатків, має API для роботи з базою та зручний і легкий й налаштуванні інтерфейс адміністратора сайту, тому саме він обраний як веб-фреймворк. PostgreSQL і MySQL є об'єктно-реляційними системами керування базами даних і надають широкий спектр можливостей для проектів Django. Але PostgreSQL має більш можливостей інтеграції з обраним фреймворком Django [7].

Виконаємо моделювання системи визначення об'єктів і наведемо діаграму варіантів її використання (рис. 1). У Web-додатку, побудованому за архітектурою клієнт-сервер, послуги надає Web-сервер, а клієнтом є браузер, або текстовий редактор, підключений до Інтернету.

Розглянувши загальну модель клієнт-серверної архітектури, перейдемо до моделювання архітек-

тури реалізованого додатка, враховуючи взаємодію між користувачем і компонентами системи (рис. 2).

Після вибору основних компонентів системи перейдемо до її реалізації. Для реалізації розбиття зображення створено декілька класів. Для початку розбиття створено клас ImageSeparator із методом виклику separate_image. Код конструктора класу наведено в лістингу 1.

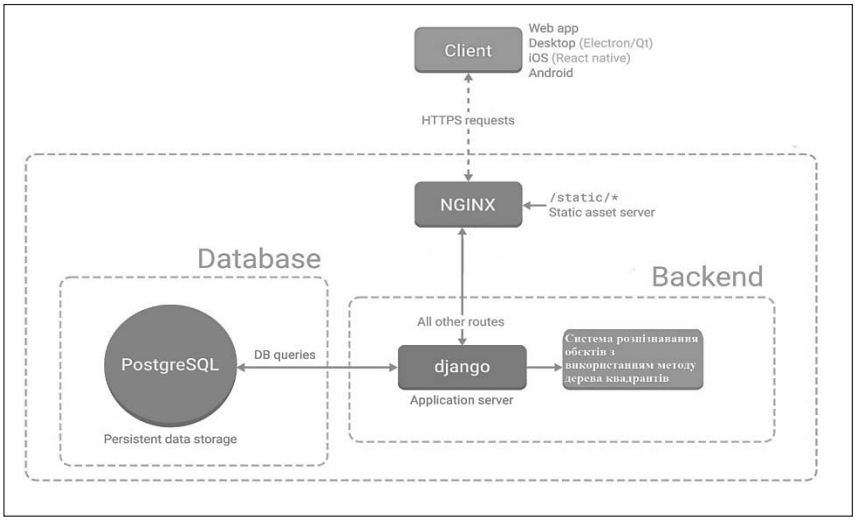

Рис. 2. Схема взаємодії компонентів системи

Лістинг 1 – Код конструктора

```
class ImageSeparator:
    def __init__(self, img_path: str,
    min_color_distance: int = 10,
    min_quad_size: int = 10):
        self.min_quad_size = min_quad_size
        self.min_color_distance = min_color_distance
        self.quad_tree_rectangles = []
        with Image.open(img_path) as img:
            self.width = img.width
            self.height = img.height
            self.pixels = img.load()
```

Як аргументи передаємо: img_path – шлях до зображення; min_color_distance – мінімальну різницю кольорів; min_quad_size – мінімальне значення розміру вузла дерева. Метод separate_image класу ImageSeparator застосовуємо для розбиття зображень.

Під час опису процесу розпаралелення обчислень розглянуто особливості CPython під час використання потоків і процесів. Він оптимізований для роботи в однопоточному режимі й має нюанси під час роботи в багатопоточному режимі, що пов'язані з GIL (global interpreter lock), який не дає декільком потокам виконувати одночасно код Python. Багатопоточне виконання займає більше часу [5]. Причини використання GIL: однопоточні сценарії виконуються швидше; проста реалізація й інтеграція бібліотек на C. У лістингу 2 наведено функцію separate_image(), що використовує процеси для розпаралелення.

Лістинг 2 – Функція separate_image()

```
def separate_image(self):
    leafes = self.create_initial_leafs(self.width, self.height)
    queue = Queue()
    proceses = []
    for leaf in leafes:
        worker = ImageSeparatorWorker(self.min_color_distance,
self.min_quad_size, self.pixels, queue)
        p  =Process(target=worker.split_rectangle, args=(leaf,))
        proceses.append(p)
        p.start()
    for p in proceses:
        self.quad_tree_rectangles.extend(
self.yield_from_process(queue, p))
    return self.quad_tree_rectangles
```

У separate_image використовується початкове розбиття зображення на 4 рівні частини [8]. Для обробки в циклі створюється клас ImageSeparatorWorker та окремий процес, якому в конструктор передається split_rectangle.

Розглянемо структуру проекту та його налаштування (рис. 3).

У проекті для роботи з базою даних використана Django ORM – техніка програмування, призначена для перетворення несумісних типів даних в об'єктна-орієнтованих мовах програмування. Для роботи з алгоритмом створено дві моделі, код наведено в лістингу 3.

Лістинг 3 – Створення моделей для роботи з базою даних

```
class ImageQuadTree(models.Model):
    image = ImageField()
    quad_tree_repr = JSONField(blank=True, null=True)
    quad_tree_image = ImageField(blank=True)
    min_color_distance = models.IntegerField(
            validators=[MinValueValidator(5)], default=7)
    min_quad_size = models.IntegerField(
            validators=[MinValueValidator(9)], default=16)
class ImageQuadTreeCompare(models.Model):
    first_image = models.ImageField()
    second_image = models.ImageField()
    quad_tree_diff_repr = JSONField(blank=True, null=True)
    quad_tree_diff_image = ImageField(blank=True)
    min_color_distance = models.IntegerField(
validators=[MinValueValidator(5)], default=7)
    min_quad_size = models.IntegerField(
validators=[MinValueValidator(9)], default=16)
```

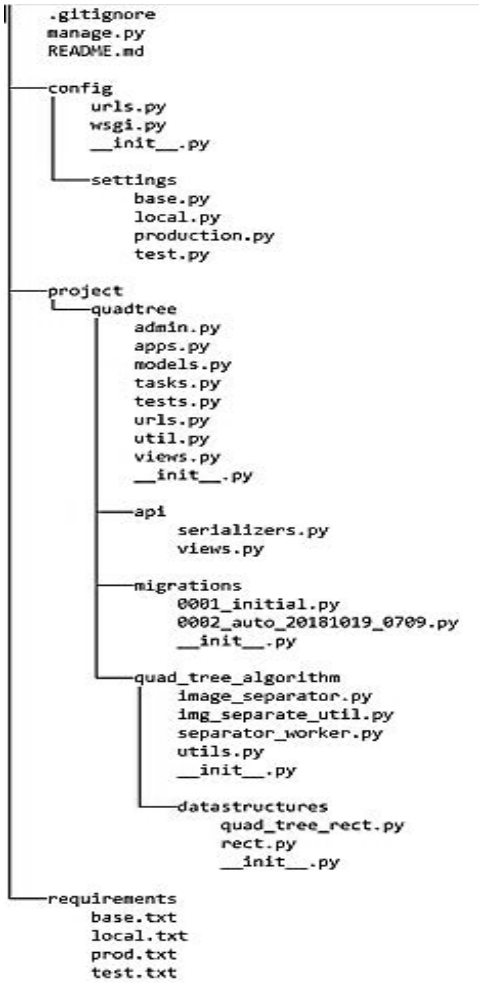

Рис. 3. Структура проекту

Після створення моделей виконуємо команди: manage.py makemigrations – створення нових міграцій на основі змін у моделях; manage.py migrate – застосування міграцій, відкат міграцій і виведення статусу міграцій. Варто розглядати міграції як систему контролю версій для бази даних. Додаток Django admin використовує моделі для автоматичного створення частини сайту, призначеної для створення, перегляду, оновлення й видалення записів у БД. Для реалізації REST системи використаємо Django Rest Framework. API DRF складається з 3-х шарів: серіалізатора, виду (view) і маршрутизатора. Серіалізатор: перетворює інформацію, що зберігається в базі даних і визначена моделями Django, у формат, який легко й ефективно передається через API. Код серіалізатора для порівняння зображень наведено на лістингу 4.

Лістинг 4 – Код серіалізатора

```
class QuadTreeCompareSerializer(serializers.ModelSerializer):
   class Meta:
      model = ImageQuadTreeCompare
      fields = '__all__'
      read_only_fields = ('min_color_distance',
 'min_quad_size',)
```

Вид визначає функції, які доступні через API. У системі реалізовано два view підкласи API, в кожному перевизначено метод post(). Користувач системи за допомогою HTTP методу POST робить запит, передавши зображення та необхідні параметри. Код view наведено в лістингу 5.

Лістинг 5 – Код view для порівняння зображень

```
class ImageQuadTreeView(APIView):
   serializer_class = QuadTreeSerializer
   def post(self, request):
      serializer = QuadTreeSerializer(data=request.data)
      if serializer.is_valid():
         quad_tree = serializer.save()
         data =process_image_quad_tree_separation(
quad_tree.id)
         return Response(serializer.to_representation(data),
 status=status.HTTP_201_CREATED)
      returnResponse(serializer.errors,
 status=status.HTTP_400_BAD_REQUEST)
```

Маршрутизатор визначає url-адреси, які надають доступ до кожного view і зберігаються у файлі url.py. Після отримання коду моделей, серіалізатора та view перейдемо до перевірки роботи системи [8; 9]. Приклад результату роботи системи наведено на рис. 4. На вхід подано 2 зображення супутникових знімків аеропорту. Для кожного зображення побудовано дерева квадрантів і на виході згенеровано результуюче зображення з відмінностями між вхідними.

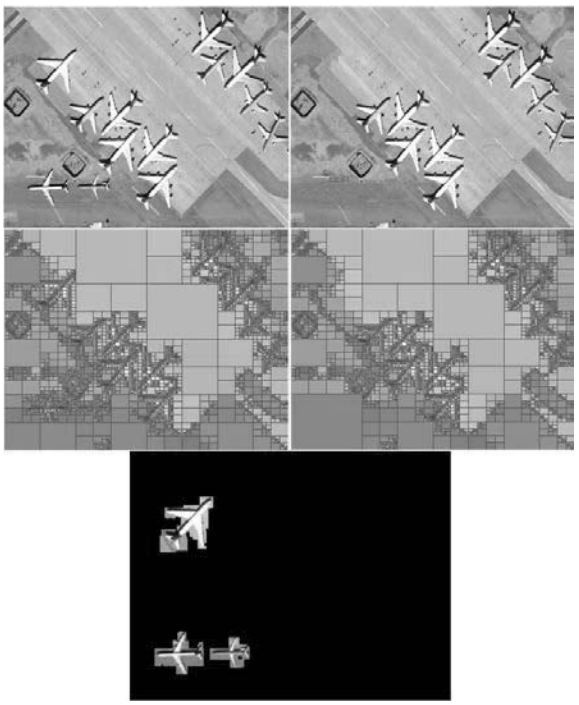

Рис. 4. Результат роботи системи

Для тестування системи використано метод написання unit-тестів. Це дає змогу швидко перевірити, чи не призвела зміна коду до появи помилок у вже протестованих місцях програми та чи полегшує усунення таких помилок. Тест, який перевіряє успіх створення об'єкта, наведено в лістингу 6.

Лістинг 6 – Тест успішного створення об'єкта

```
def test_split_valid_image(self):
   response = self.client.post(
      reverse('split'),
      data=self.valid_payload,
      format='multipart')
   self.assertEqual(response.status_code,
   status.HTTP_201_CREATED  )
```

Тест для перевірки правильної роботи системи, коли користувач передав невалідні дані на вхід, наведено в лістингу 7.

Лістинг 7 – Тест коректної відповіді в разі невалідних даних

```
def test_split_invalid_image(self):
      response = self.client.post(
         reverse('split'),
         data=self.invalid_payload,
         format='multipart'    )
      self.assertEqual(response.status_code,
   status.HTTP_400_BAD_REQUEST  )
```

Так створено набір unit-тестів для перевірки функціоналу системи, який полегшує внесення змін до проекту й перевірку їх на наявність помилок.

У розроблюваній системі існують ділянки коду, які виконуються як послідовно, так і паралельно. На лістингу 8 зображено створення процесів, їх запуск і після завершення їх роботи зчитування результатів.

Лістинг 8 – Розпаралелювання обчислень

```
queue = Queue()
proceses = []
for leaf in leafes:
   worker = ImageSeparatorWorker(self.min_color_distance,
 self.min_quad_size, self.pixels, queue)
   p = Process(target=worker.split_rectangle,args=(leaf,))
   proceses.append(p)
   p.start()
for p in proceses:
   self.quad_tree_rectangles.extend(
self.yield_from_process(queue, p))
```

Як видно на лістингу, в коді існують ділянки коду, що виконуються паралельно: розбиття ділянок зображення на дерева квадрантів; так само існують ділянки коду, які виконуються й послідовно – отримання даних із процесів, створення екземпляру класу Queue.

Кількість створюваних процесів залежить від кількості ділянок зображення, які треба розбити. Залежність між збільшенням числа процесів і швидкодією системи близька до лінійної [8]. Прискоренням паралельного алгоритму є відношення часу виконання кращого послідовного алгоритму до часу виконання паралельного алгоритму: $S = T_1/T_p$. Отже, прискорення для двох процесів дорівнює 11/6,92 = 1,58, а для чотирьох – 11/3,78 = 2,91.

Висновки. У роботі розроблено метод порівняння зображень на предмет їх відмінностей шляхом порівняння їх дерев квадрантів; удосконалено метод визначення із заданою точністю ділянок зображень, що відрізняються, способом отримання частини зображення у вигляді квадрантів, які включають у себе координати й колір частини зображення. Авторами впроваджена клієнт-серверна система визначення об'єктів з використанням методу дерева квадрантів. У ході подальших досліджень планується вдосконалення розроблених методів.

Список літератури:

1. Новейшие методы обработки изображений / А.А. Потапов, Ю.В. Гуляев, С.А. Никитов, А.А. Пахомов, В.А. Герман. Москва: ФИЗМАТЛИТ, 2008. 496 с.
2. Zaitoun N.M., Aqel M.J. Survey on Image Segmentation Techniques. *Procedia Computer Science.* 2015. Vol. 65. P. 797–806.
3. Spann M., Wilson R. A quad-tree approach to image segmentation which combines statistical and spatial information. 1985. Vol. 18 (issue 3–4). P. 257–269.
4. Hunter G.M., Steiglitz K. Operations on Images Using Quad Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 1979. Vol. PAMI-1 (issue 2). P. 145–153.
5. Dierbach Ch. Python as a first programming language. *Journal of Computing Sciences in Colleges.* 2014. Vol. 29 (issue 6). P. 153–154.
6. Greenfeld D., Greenfeld A. Two Scoops of Django: Best Practices for Django 1.8. 3rd ed. Chicago: Two Scoops Press, 2015.
7. Васильев А.Ю. Работа с PostgreSQL: настройка и масштабирование. 2017. URL: https://postgresql.leopard.in.ua/ (дата звернення: 14.01.2019).
8. Kirichek G., Kurai V. Implementation quadtree method for comparison of images. Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). 2018. Lviv-Slavske. Ukraine. February 20–24. P. 129–132.
9. Киричек Г.Г., Курай В.І. Визначення об'єктів з використанням дерева квадрантів. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка».* 2018. Вип. 1 (26). С. 19–24.

КЛИЕНТ-СЕРВЕРНАЯ СИСТЕМА ОПРЕДЕЛЕНИЯ ОБЪЕКТОВ

В работе рассмотрены вопросы разбиения и сравнения изображений с целью реализации высокоэффективного доступа к элементам. Для достижения поставленной цели используется рекурсивное разбиение двумерного пространства, работа с клиент-серверной архитектурой и фреймворком Django.

Ключевые слова: *api, сегментация, распознавание объектов, сервер, клиент.*

CLIENT-SERVER SYSTEM OF DETERMINATION OBJECTS

The paper deals with the question of the partitioning and comparison of images in order to implement highly effective access to the elements. To achieve this goal used recursive partitioning of the two-dimensional space, work with the client-server architecture and with the framework Django.

Key words: *api, segmentation, object recognition, server, client.*