Демонстрационный вариант
задания заключительного этапа
по направлению «Программная инженерия»

Категория участия: «Магистратура/специалитет»

Задание 1. Продажи строящегося жилья (максимум 30 баллов).

С 2019 года работает новая схема продажи строящегося жилья с использованием эскроу-счета. Эскроу-счет, который открывает застройщик – это специальный банковский счет, на котором размещаются денежные средства покупателей жилья до момента окончания строительства. Ни покупатель жилья, ни застройщик не могут использовать их до передачи готового жилья или расторжения договора долевого участия.

Некий банк занимается доработкой системы, которая позволит работать с эскроу-счетами. С ней будут работать операторы (вводят данные в систему) и аудиторы (получают данные из системы).

Необходимо, разработать структуру реляционной базы данных, которая бы позволила хранить данные об эскроу-счетах и покупателях жилья, а также существенные условия договора эскроу. Основные реквизиты покупателей: ФИО для физического лица или наименование для юридического лица, серия, номер, и дата выдачи паспорта для физического лица или ОГРН для юридического лица, адрес местонахождения и почтовый адрес покупателя.

Необходимо:

1) Используя любую общепринятую нотацию, изобразить схему инфологической модели предметной области.

2) Используя любую общепринятую нотацию, изобразить схему даталогической модели базы данных, удовлетворяющую третьей нормальной форме, с выделением первичных и внешних ключей, типа и направления связей.

3) Изобразить прототипы web-страниц работы пользователей с ролями «Оператор» и «Аудитор» с системой. Прототипы должны иллюстрировать возможности ввода и вывода доступной пользователям информации.

4) Описать алгоритм, который автоматически вернет деньги на счет покупателя жилья, если сдача дома задерживается более, чем на полгода.

5) Используя операторы языка SQL, написать запрос, который позволит аудитору получить список покупателей жилья для конкретного застройщика с указанием внесенных ими денежных сумм.

Требования к структуре оформления решения с указанием критериев оценивания и максимального количества баллов за каждую часть решения:

1. Введение – до 2 баллов. 2 балла дается в случае, если введение содержит кроме переписывания задания общую идею решения.
2. Основная часть:

2.1. Схема решения, описание процессов, описание инфологической модели предметной области – до 3 баллов

2.2. Структура базы данных, отражающая специфику предметной области. БД должна соответствовать третьей нормальной форме, не быть перегруженной дублированием, учитывать сущности-справочники, не иметь ошибок по связям – до 5 баллов.

2.3. Прототипы визуальных интерфейсов – до 6 баллов.

2.4. Описание алгоритма (в виде схемы алгоритма, программного кода или псевдокода) – до 6 баллов.

2.5. Описание SQL-запроса, подготовка скрипта на создание таблиц и вставку данных, необходимых для проверки результатов выполнения запроса – до 6 баллов.

3. Заключение (выводы) – до 2 баллов. 2 балла дается в случае, если в заключении перечислены все результаты работы.

Задание 2. Размещение виртуальных машин на вычислительном кластере (максимум 30 баллов).

Для организации сервиса облачных вычислений необходимо предложить проект системы, обеспечивающей запуск виртуальных машин на масштабируемом вычислительном кластере.

Исходные данные:

- имеется парк серверов (N штук, 10^3 < N < 10^5), объединенных в кластер общей сетью. На каждом сервере -- стандартная ОС и одинаковое количество вычислительных ядер и оперативной памяти (128 Gb);

- существующий инструмент (программа), умеющая запускать виртуальную машину с заданным размером памяти и вычислительным заданием ("задание", оно же "вычислительное задание" -- это полезная нагрузка, которую может исполнять виртуальная машина. Например, решение какого-то численного уравнения. Не путать с запросом к API);

- поток случайных запросов на создание виртуальных машин с параметрами:

{"id": <unique machine id, numeric>, "size": <RAM, Gb, numeric>, "task": <payload, string>}, где size - может принимать значения степеней 2 (1, 2, 4 ... 128).

Необходимо:

1. Спроектировать и описать систему, предоставляющую API по созданию виртуальных машин, а также обеспечивающую оптимальное размещение машин на кластере и передачу файлов заданий на сервера.

2. API должно принимать запросы в описанном формате и отвечать пользователю:

- {"result": "OK", "host_id": <nmb of target host in cluster, 1...N, numeric>}, где nmb -- номер сервера, на который размещена виртуальная машина) в случае возможности разместить машину,

- {"result": "NOT_OK"}, в случае невозможности разместить машину с такими требованиями по RAM.

3. Описать алгоритм размещения виртуальных машин на кластере. Алгоритм должен обеспечивать оптимальное размещение машин на кластере для оптимальной утилизации памяти. "Утилизация" означает среднюю "степень загруженности" **используемых серверов** - чем она выше, тем меньше неиспользуемой RAM на сервере. утилизация в 1 означает что все ресурсы сервера используются (нет свободных), а утилизация в 0.75 означает что на сервере есть 1/4 свободной памяти, т.е. 32 Gb;

4. Реализовать алгоритм в отдельном микросервисе с аналогичным API для автоматической верификации алгоритма.

Требования к структуре оформления решения с указанием критериев оценивания и максимального количества баллов за каждую часть решения:

1. Введение. Описание архитектуры предлагаемой системы с основными компонентами их взаимодействием – до 2 баллов.
2. Основная часть.

   2.1. Выбор критерия оценки эффективности размещения машин на кластере – до 3 баллов.

   2.2. Обоснование и описание алгоритма размещения машин на кластере – до 3 баллов.

   2.3. Описание в форме UML-диаграмм моделей процессов: хранения и обработки запросов на размещение виртуальных машин, обеспечения отказоустойчивости кластера при типовых отказах (сервера, процесса выполняемого задания) – до 3 баллов.

   2.4. Разработка прототипа для API размещения машин на кластере серверов и тестирование прототипа API на функциональность с представлением результатов теста исходных кодов, опубликованных в любом репозитории, работающим с VCS git с разрешением публичного доступа, и инструкций по воспроизведению проверяющими экспертами, и содержащий Dockerfile, полностью реализующий сборку и запуск прототипа таким образом, чтобы запущенный из этого Dockerfile контейнер предоставлял требуемый API на порту 9024 – до 15 баллов.

   2.5. Выбор параметров для сбора статистики и оценки эффективности использования серверов в кластере, создание сценариев масштабирования (добавления и сокращения серверов) кластера – до 3 баллов.
3. Заключение (выводы по достижению поставленных целей) – до 1 баллов.

Задание 3. Система компьютерного зрения для распознавания дефектов дорожного покрытия (максимум 40 баллов)

Предложить алгоритмы и программное обеспечение в сфере компьютерного зрения для обнаружения дефектов дорожного покрытия на видео, снятого с камеры, установленной в автомобиле.

Пример исходных данных:

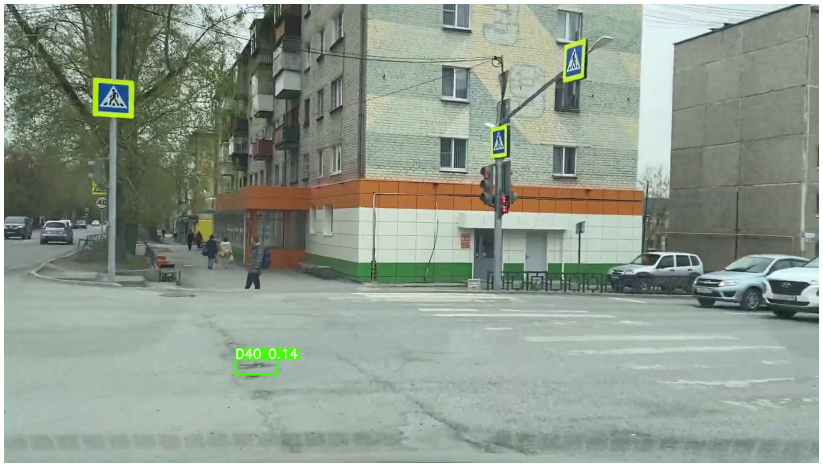
В работе допустимо применять любые opensource библиотеки: OpenCV, PyTorch и другие. Допускается использование любых известных на сегодняшний день архитектур нейронных сетей. К программному обеспечению предъявляются умеренно жесткие требования по быстродействию (до 20 мс на распознавание одного изображения на видеокарте уровня NVidia GeForce 1050 Ti). Кроме того, решение должно быть построено на основе применения библиотек NVidia CUDA, то есть обязательно возможность применения на видеокартах NVidia. Рекомендуется использование языка Python или C++, но не обязательно.

Классы дефектов:

1. Яма (выбоина) — обязательно.
2. Дорожный люк (для дифференциации от выбоин) – обязательно.
3. Трещина (продольная/поперечная) — опционально.
4. Сетчатые трещины — опционально.
5. Стертая разметка — опционально.

Перед определением наличия дефектов на дорожном покрытии настоятельно рекомендуется применить преобразование изображения: выделение области дороги, геометрические и цветовые преобразования.

Необходимо также приложить исходный код в виде архива и выгрузить его в любой git-репозиторий (github, gitlab, bitbucket) с предоставлением публичного всеобщего доступа и приложить ссылку. Пример распознавания дефекта:



Критерии оценивания:
Проверка работоспособности на тестовом наборе данных с помощью метрик precision, recall, mean average precision (mAP).

Наборы данных доступны по ссылкам:

https://disk.yandex.ru/d/5x1A_vuz6T0T5A

https://disk.yandex.ru/d/6J6OIH8HB9Z9Dg

Требования к структуре оформления решения с указанием критериев оценивания и максимального количества баллов за каждую часть решения:

1. Введение (изложить главную идею решения) - до 1 балла.
2. Основная часть:

2.1. Описание технологии обнаружения с кратким описанием методов компьютерного зрения, машинного обучения, предварительной обработки данных используемых для этой цели. – до 5 баллов.

2.2. Описание алгоритма преобразования и подготовки изображений - до 8 баллов.

2.3. Описание программного обеспечения, используемого для получения и использования моделей машинного обучения – до 6 баллов.

2.4. Программная реализация решения – до 9 баллов.

2.5. Демонстрация решения на тестовых примерах (для всех классов дефектов, не менее 3-х различных изображений для каждого класса дефекта) с использованием программной реализации – до 7 баллов. В случае демонстрации решения не для всех классов дефектов – до 1 балла за 1 класс дефекта.

2.6. Проверка работоспособности на наборах данных с помощью метрик precision, recall, mean average precision (mAP) – до 3 баллов.

3. Заключение (выводы) – до 1 балла.