ИДЗ - состав, порядок работы

Порядок выполнения

Для того, чтобы минимизировать риски неудачного создания пояснительной записки/приложения в последний момент, предлагается следующая последовательность шагов:

Как работать в репозиториях

Все требования по работе в репо перечислены в описаниях заданий ниже. Создавать специально для меня PR НЕ НУЖНО. Ждать моей проверки на ваших PR - бессмысленно:(

Я проверяю только содержимое ветки main - прошу к моменту сдачи заданий мержить наработки в main.

Как сдавать задания

На всякий случай, как оказалось, не все этот момент увидели в списке ниже - **все задания по курсу сдаются путем написания письма преподавателю**. Как и что писать - указано ниже.

- Команда внимательно читает требования ниже, убеждается, что их результат соответствует описанию.
- Команда внимательно смотрит в README своего репозитория и убеждается, что там напротив сдаваемого задания (все кроме 0) стоит метка Passing зеленого цвета.
- Один участник команды (кого команда уполномочила сдавать это задание) пишет письмо преподвателю с темой **“[NoSQL] Название_задания - номер_И_название_проекта”**.
- Остальные участники указываются в копии письма.
- Переписка по одному заданию остается в рамках одной цепочки писем.
- Если вам необходимо указать на объект в репозитории, предоставьте полную ссылку на него (такую, которая откроется по нажатию в браузере).
- Если вы отправляете скринкаст - присылайте ссылку на него (прикладывать к письму не нужно).

Желательно, чтобы все участники команды сдали примерно одинаковое количество заданий ИДЗ (== этапов, описанных ниже).

Что такое массовый импорт-экспорт

В каждой из ИДЗ необходимо реализовать функциональность массового импорта-экспорта. Что она подразумевает:

1. возможность импорта экспорта всех данных из системы в машино-читаемом формате

(json, xml, csv …. на ваш выбор)
2. пользовательские интерфейсы для импорта и экспорта

Зачем это нужно:

1. потренироваться в создании простого модуля бакапа
2. посмотреть, какие возможности предоставляет ваша СУБД для работы с дампами БД

Согласована и сформулирована тема курсовой

1. Выбрать тему и согласовать ее с преподавателем. Вам нужно письмом:
    1. Предложить свое виденье того, как будет устроено приложение.
    2. Озвучить используемые инструменты (язык программирования и БД), либо посоветоваться с преподавателем.
    3. Получить (и проверить) доступ в репозиторий

Установка и настройка выбранной БД + ЯП

Смысл данного задания - сделать минимально работоспособный пример (скрипт для командной строки) работы “выбранной БД + ЯП” на выбранном вами языке программирования, продемонстрировать его работоспособность в виде скринкаста и залить в репо. Для того, чтобы исключить на поздних этапах проекта проблемы совместимости.

Скрипт может быть быть просто HelloWorld - не связанным с ИДЗ.

1. Результат:
    1. На вашем компьютере можно создать и запустить пример (см выше) для выбранной вами пары “Язык программирование” - БД. Приложение должно подключится к БД, записать и прочитать данные из нее.
    2. Работа приложения из пункта выше демонстрируется в виде скринкаста (скринкаст == короткая запись **видео** с вашего экрана). Не заливайте его в репо (можно с гугл / яндекс диска)
    3. Код приложения выше загружен в репозиторий, в ветку **main**, каталог **hello_world**.
    4. Делать для этого PR не нужно.
    5. Docker на данном этапе не нужен.

Use case

Презентация про то, как составлять макет и писать сценарии использования (+типичные ошибки)

Результат размещен на вики в виде **одной** страницы “Макет и сценарий использования”:

1. У вас есть формулировки основных сценариев использования приложения (текст, размещенный на вики, а не ссылка на документ) и макета его пользовательского интерфейса (большое изображение), **размещенные на вики вашего репозитория**.
2. Макет изображен в виде графа

(https://hsto.org/files/026/cab/55a/026cab55a5ad4a1daab209c39715b947.png).
3. Файл макета загружен в репозиторий под названием ui_mockup.png (jpg, svg), макет вставлен как изображение на вики.
4. Каждый экран на макете имеет или уникальное осмысленное название или номер, по которому его можно идентифицировать.
    1. Дублировать экран на макете не нужно - если есть какое-то поведение, которое вы хотите показать и для этого нужно дублирование экрана, лучше опишите его в сценарии использования.
5. Все надписи и элементы на макете различимы и читаемы.
6. Макет включает в себя элементы управления для массового импорта/экспорта данных и построения кастомизируемой статистики.
7. Вы знаете какие из сценариев использования вы сможете продемонстрировать в рамках прототипа.

На что стоит обратить внимание при составлении макета (типовые ошибки):

- Если у вас в макете фигурирует какая-то сущность (пользователь, поставщик, автомобиль, животное, здание ...), которая фигурирует во множественном числе, у нее обязательно должна быть
    - страница для просмотра одного элемента (страница пользователя, поставщика ...),
        - если элемент подразумевает редакитрование, то надо на этой же странице добавить даты создания и редактирования элемента,
        - почти всегда будет в кассу дополнять такую страницу полем “Комментарий”, где пользователи смогут хранить произвольную информацию, которая не укладывается в модель (но это не приглашение использовать это поле в качестве мусорки),
    - страница поиска элементов по всем полям (поиск пользователей, поставщиков ...) и отображение результатов в виде таблицы .
- Если логика вашего приложения подразумевает какие-либо процессы (например - трекинг задач у программистов, почтовые отправления, заказы в Интернет-магазине) или хотя бы переменные статусы (например - у сотрудника могут быть статусы Уволен, Уволился сам, В отпуске, Работает, Сокращен,), то важно в макете предсмотреть просмотр и поиск по истории изменения состояний статуса с привязкой ко времени.
- Как правило, в большинстве проектов нужны интерфейсы поиска сразу по нескольким сущностям (коллекциям) - например, все курьеры, на которых есть N незакрытых заказов,

Модель данных

1. Подготовить страницу на вики репозитория под названием “Модель данных”. Результат (на вики):
    1. На вики странице есть два раздела “Нереляционная модель” “Реляционная модель” “Сравнение моделей”, “Вывод”
        1. Нереляционная модель (для вашей СУБД)
            1. Графическое представление модели (сущности и связи, типы данных, коллекции - можно использовать ER-диаграмму, можно json-схему, если она применима к вашему типу СУБД)
            2. Описание назначений коллекций, типов данных и сущностей
            3. Оценка объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества

объектов - нужно выразить через переменную (количество одного из видов объектов вашей БД)). У вас должна получится формула - зависимость объема от одной переменной.
4. **Избыточность модели (отношение между фактическим объемом модели и “чистым” объемом данных).**. У вас должна получится формула - зависимость избыточности от одной переменной.
5. **Направление роста модели при увеличении количества объектов каждой сущности.**
6. Запросы к модели, с помощью которых реализуются сценарии использования
    1. Текст запросов
    2. Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров
    3. Количество задействованных коллекций (если есть)

2. “Реляционная модель” - Аналог модели данных для SQL СУБД - характеризуется аналогично нереляционной (см. подпункты выше)
3. “Сравнение моделей”
    1. Удельный объем информации (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов) - где данные занимают больший объем при прочих равных (http://se.moevm.info/doku.php/staff:courses:no_sql_introduction:calculating_data_model_size)
    2. Запросы по отдельным юзкейсам:
        1. Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров
        2. Количество задействованных коллекций (если есть)
    3. “Вывод” - что лучше, SQL или NoSQL модель
2. Есть примеры хранения данных в БД.

Прототип*

1. Подготовить прототипы приложения (“Хранение и представление” и “Анализ”). Результат:
    1. Приложение компилируется и реализует оговоренные сценарии использования **через пользовательский интерфейс**.
    2. Исходники и исполняемые файлы собранного приложения лежат в репозитории, там поставлен тег 0.5 (0.8). Что такое тег - https://git-scm.com/book/ru/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D1%82%D0%B5%D0%B3%D0%B0%D0%BC%D0%B8

Про пользовательский интерфейс, роли и пользователей:

1. Пожалуйста, не создавайте несколько разных и несвязанных интерфейсов (на разных контейнерах вашей системы) - сделайте один интерфейс для всех ролей и пользователей.
2. В первую очередь реализуйте авторизацию через логин и пароль, затем (если у вас есть такое желание - не обязательно) - иными способами.

Что подразумевает **прототип “Хранение и представление”** -

1. приложение запускается,
2. позволяет просматривать содержимое БД с помощью таблиц / списков
3. выполняет одну из следующих функций: или позволяет добавление новых элементов данных в БД, или предоставляет интерфейсы для поиска (фильтрации) данных,
4. если в приложении есть авторизация, то в нем должны по умолчанию присутствовать по одному отладочному пользователю на каждую роль, сами пользователи и их данные указаны в README
5. **вы можете продемонстрировать выполнение всех пунктов выше скринкастом (не более двух минут - сключите запись процесса компиляции / загрузки и тд) или работоспособного процесса сборки из репо через docker-compose**

Что подразумевает **прототип “Анализ”** - выполненны требования “Хранение и представление”

1. **приложение разворачивается через docker-compose build –no-cache из репозитория на ubuntu 22.04+**,
2. сервер вашей СУБД добавлен docker-compose.yml (== вы не используете БД извне ваших контейнеров, у вас есть контейнер db), типовые ошибки по данному пункту:
    1. не выставляйте порт БД наружу
    2. дополняйте маппинг портов указанием локального интерфейса везде, например '127.0.0.1:8081:8081'
    3. не монтируйте локальные каталоги, монтируйте volume (если вам нужны исходники / файлы проекта, то лучше копируйте их на этапе сборки образа)
    4. для контейнера СУБД (за исключением memcached и других хранилищ в памяти) обязательно должен использоватся volume, куда будет монитровatся директория с данным СУБД
    5. вместо тега latest указывайте конкретный тег (версию) для образов
    6. не выности мапинг портов в переменные среды - в этом нет необходимости, его можно захардкодить вполне
3. добавлена недостающая часть (либо добавление элементов, либо поиск),
    1. поиск элементов реализован следующим образом - для каждого поля сущности есть отдельное поле (или поля) ввода поискового запроса, что позволяет искать по сложным запросам, например “найди всех сотрудников, у которых пол мужской, дата рождения до 15.01.2002, имя - Олег”. Ориентируйтесь на подробные фильтры в интернет-магазинах.
    2. как нелья реализовывать поиск по нескольким атрибутам: в виде одного поля ввода (input) и выпадающего списка / radiobutton с выбором поля, для которого будет использоваться введенный текст
    3. подумайте, а что вам за ваш фильтр скажут на собеседовании / на текущей работе? Если на ум приходят нецензурные слова, возможно стоит доработать механизм фильтрации.
4. добавлен импорт и экспорт всех данных приложения в машиночитаемом формате (XML, JSON, CSV ….) одним действием (одна кнопка Импорт импортирует данные сразу для всего приложения, одна кнопка Экспорт экспортирует сразу все данные всего приложения), отдельные кнопки для отдельных сущностей делать не нужно.
5. в БД приложения содержится достаточный набор тестовых данных для демонстрации всех реализованных сценариев использования,

App is ready

1. Продемонстрировать работоспособность всех сценариев использования на окончательной версии приложения.
    1. Приложение компилируется и реализует все сценарии использования.
    2. В приложении есть к представлению данных в виде таблиц добавилась serverside пагинация.
    3. В приложение добавлено вычисление и отображение статистики / анализа данных / необходимых вычислений согласно заданию.
    4. Вы можете продемонстрировать это через docker-compose + дистанционно / с помощью скринкаста.
    5. Исходники и исполняемые файлы собранного приложения лежат в репозитории, там поставлен тег 1.0.

Пояснительная записка.

Пояснительную записку можно писать из любого состояния проекта (не обязательно доводить его до полностью готовой версии). Пример - вы сделали задания на Удовлетворительно, соответственно, пишите в записке о том, что у вас получилось.

1. Предоставить пояснительную записку в электронном виде. Результат:
    1. Ваша пояснительная записка полностью соответствует требованиям к оформлению (http://eltech.ru/assets/files/3004_3_ShABLON-kursovika.doc)и содержанию (описание)
    2. Записка выложена в репозиторий в редактируемом формате и pdf (в корне репозитория созданы файлы report.pdf и report.odt / report.docx).

Пример хорошей записки
https://github.com/moevm/nosql-2017-lib_card/blob/master/%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%BA%D0%B0.pdf

Состав и оформление пояснительной записки

Создайте в корне репозитория два файла report.docx (doc, odt) и report.pdf.

В отчет можно компилировать тексты из предыдущих заданий (но, пожалуйста, следите за их оформлением:).

Для оформления вам пригодятся следующие ссылки:

- Шаблон оформления пояснительной записки к ИДЗ (отчета)
- Сервис для конвертации markdown (формат вики Github) в docx

Разделы ИДЗ:

1. Введение
    1. Актуальность решаемой проблемы

    2. Постановка задачи
    3. Предлагаемое решение
    4. Качественные требования к решению
2. Сценарии использования
    1. Макет UI
    2. Сценарии использования для задачи
        1. импорта данных:
            1. как пользователь загружает данные в программу массово (импорт из файла или внешнего источника)
            2. как пользователь загружает данные в программу вручную
        2. представления данных
            1. как данные отображаются для пользователя и что он должен сделать для их отображения (поиск, визуализация в виде графиков и таблиц)
        3. анализа данных
            1. какие действия должен сделать пользователь для проведения анализа данных (подсчет средних, статистик и тд)
        4. экспорта данных
            1. как пользователю получить копию хранимых в программе данных в машиночитаемом формате (JSON, XML, CSV ….)
    3. Вывод о том, какие операции (чтение или запись) будут преобладать для вашего решения.
3. Модель данных
    1. Нереляционная модель данных (для вашей СУБД)
        1. Графическое представление
        2. Описание назначений коллекций, типов данных и сущностей
        3. Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов)
        4. Запросы к модели, с помощью которых реализуются сценарии использования
    2. Аналог модели данных для SQL СУБД - характеризуется аналогично нереляционной (см. подпункты выше)
    3. Сравнение моделей
        1. Удельный объем информации - где данные занимают больший объем при прочих равных (http://se.moevm.info/doku.php/staff:courses:no_sql_introduction:calculating_data_model_size)
        2. Запросы по отдельным юзкейсам:
            1. Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров
            2. Количество задействованных коллекций (если есть)
            3. Сложность запроса
        3. Вывод - что лучше, SQL или NoSQL модель
4. Разработанное приложение
    1. Краткое описание (из каких модулей / контейнеров состоит, какую архитектуру вы использовали)
    2. Использованные технологии
    3. Снимки экрана приложения
5. Выводы
    1. Достигнутые результаты
    2. Недостатки и пути для улучшения полученного решения
    3. Будущее развитие решения

6. Приложения
    1. Документация по сборке и развертыванию приложения
    2. Инструкция для пользователя
7. Литература (обязательно должна быть ссылка на ваш репо)

Возможные вопросы в процессе сдачи

1. Термины из курса и их понимание
2. Обоснования решений из ИДЗ
3. Сравнение с SQL базами данных

Top secret

- Mongo
- Neo4j
- Memcached
- ArangoDB
- Cassandra
- DGraph