Задания к курсу
“Методы обработки и хранения данных”
2023-2024 уч. г.

1. Реализуйте функционал для игры в фризские шашки (вариант 5x5 - Frysk!). Необходимые контракты (интерфейсы) и их реализации (фигуры, доска, ходы, нотация и т. д.) в соответствии с правилами, продумайте самостоятельно. О правилах игры можно прочитать здесь:
   https://lidraughts.org/variant/standard
   https://lidraughts.org/variant/frisian
   https://lidraughts.org/variant/frysk

2. Реализуйте асинхронный сервер-посредник для отправки запросов в базу данных, хранящую информацию об игроках. База данных должна содержать следующие данные:
   - логины и “солёные” логином хэши паролей (хэш-функцию выберите самостоятельно) игроков с индексированием по логину;
   - [опционально] информацию для восстановления пароля (контрольный вопрос (один из предложенных или свой, хэш ответа на вопрос));
   - [опционально] информацию об игроках: дату регистрации, общее время активности на сервере с момента регистрации, дату последней активности;
   - [опционально] информацию о блокировке со стороны администрации игрока (срочной и бессрочной) с указанием причины.

   Схему базы данных определите самостоятельно. Все таблицы в базе данных должны находиться в третьей нормальной форме (или выше). Структуру запросов к сервису определите самостоятельно (см. другие задания). Информацию о подключении к базе данных получайте из файла конфигураций (предусмотрите трансформации для конфигураций).

3. На базе технологии gRPC или подхода REST API реализуйте асинхронный централизованный сервер, обслуживающий партии игроков в фризские шашки. Зависимые сервисы для компонентов получайте при помощи внедрения через конструктор/setter на основе выбранной Вами реализации DI/IoC (Вы можете использовать DryIoC, IServiceProvider, Autofac, Windsor Castle, Unity, etc.). Предусмотрите следующие запросы:
   - авторизация игрока по логину и паролю (запрещено передавать пароль в открытом виде) ([опционально] с уведомлением о блокировке со стороны администрации);

- создание новой игры:
    - функционал по умолчанию: авторизовавшись, игрок ждёт другого игрока (либо присоединяется к уже ожидающему игроку), после чего начинается новая партия;
    - [опционально] реализуйте функционал игрового лобби (с отображением лобби и взаимодействием с ним на стороне клиента);
- выполнение игроком хода с его сохранением на сервере в историю ходов партии и с уведомлением о ходе клиентского приложения соперника с указанием дополнительной информации (например, игра завершена);
- сдача игроком партии по его запросу с уведомлением об этом клиентского приложения соперника;
- предложение игроком ничьей по его запросу с уведомлением об этом клиентского приложения соперника;
- принятие/непринятие игроком предложенной соперником ничьей с уведомлением об этом клиентского приложения соперника и с завершением при этом партии;
- [опционально] отмена партии (возможна только до выполнения игроками первых двух полуходов - обеспечьте проверку);
- [опционально] отключение игрока посредством завершения его клиентского приложения с автоматическим поражением и уведомлением соперника о победе;
- [опционально] отправка текстового сообщения в игровой чат партии;
- [опционально] ведите лог-файл с указанием подробной информации о событиях, происходящих во время работы сервера (логи должны быть сгруппированы в разные файлы по дате записи лога).

Уведомление клиентов производите через инстанс клиентского сервиса (gRPC, REST API), полученного из DI-контейнера.

4. Реализуйте клиентское приложение (WEB или Desktop) для асинхронного взаимодействия с централизованным сервером, реализованным в задании 3. Клиентское приложение должно предоставлять следующий функционал:
    - регистрация нового игрока (пара логин/пароль, с обеспечением уникальности логина);
    - авторизация игрока по логину/паролю ([опционально] с уведомлением о блокировке со стороны администрации);
    - [опционально] восстановление пароля игрока посредством ответа на контрольный вопрос и указанием нового пароля;
    - присоединение к игре:
        - функционал по умолчанию: присоединение к игре по нажатию кнопки;

    - [опционально] создание новой “комнаты” в лобби, либо присоединение к уже существующей “комнате” (отображайте логины игроков в “комнате”); также предусмотрите функционал по отображению/скрытию заполненных в лобби “комнат”;
- во время игры:
    - [опционально] запрос на отмену партии посредством нажатия кнопки (кнопка визуализирована и доступна только до выполнения первых двух полуходов в партии);
    - выполнение игроком хода посредством взаимодействия игрока с графическим интерфейсом ([опционально] drag&drop);
    - [опционально] отображение возможных ходов в соответствии с правилами;
    - запрос на сдачу партии;
    - предложение сопернику ничьей;
    - принятие или непринятие предложенной соперником ничьей;
    - [опционально] элемент управления, отображающий полуходы игроков в партии (запись полуходов должна быть минимизирована и однозначно декодируема из позиции до выполнения конкретного полухода);
        - [опционально] возможность перехода на позицию, получившуюся после конкретного полухода при нажатии ЛКМ на соответствующее отображение модели полухода;
    - [опционально] элемент управления, отображающий текстовые сообщения из чата партии и позволяющий отправить в чат сообщение;
- [опционально] при наличии лобби, реализуйте общий чат игроков, ожидающих в лобби;
- [опционально ТОЛЬКО при наличии фичи drag&drop] Приложение должно быть реализовано на базе архитектуры MVC/MVP/MVVM/MVPVM;
- [опционально] ведите лог-файл с указанием подробной информации о событиях, происходящих во время работы клиентского приложения (логи должны быть сгруппированы в разные файлы по дате записи лога);
- объекты отображений (View), моделей отображений (ViewModel), сервисов, необходимых для клиентского приложения, должны быть получены из глобально доступного DI-контейнера;
- Предусмотрите отдельный сервис-посредник для отправки запросов по gRPC/REST API.

5. Доработайте функционал сервера из задания 3 следующим образом:
    - Реализуйте механизмы бинарных [де]сериализации для партий;
    - Вынесите хранение активных партий со стороны сервера в IMC (Redis, Apache Ignite, etc.) (ключ партии собирайте относительно логинов игроков и даты/времени начала партии) при помощи реализованных механизмов [де]сериализации; хранение информации о партиях в рамках инстанса сервера больше недопустимо;
    - по мере протекания партий обновляйте их состояние в Redis;
    - по завершении партии удалите её сериализованное состояние из Redis.

    Для подключения к IMC реализуйте сервис-фабрику, создающую объекты подключений/контекстов взаимодействия. Фабрика должна быть доступна из сервера и должна быть получена им из DI-контейнера.

6. Реализуйте асинхронный сервер-посредник для отправки запросов в базу данных, хранящую информацию о сыгранных партиях. База данных должна содержать следующие данные:
    - сериализованное состояние партии;
    - информацию об игроках, сыгравших партию;
    - дата и время начала/окончания игры.

    Схему базы данных определите самостоятельно. Все таблицы в базе данных должны находиться в третьей+ нормальной форме. Структуру запросов к сервису определите самостоятельно (см. другие задания). Информацию о подключении к базе данных получайте из файла конфигураций (предусмотрите трансформации для конфигураций).

7. Доработайте функционал заданий 5 и 6 таким образом, чтобы после удаления сериализованного состояния завершённой партии из IMC, она была сохранена в базу данных сыгранных партий. Отправку запроса на сохранение партии реализуйте через брокер сообщений (RabbitMQ, Apache Kafka, etc.). Информацию о подключении к брокеру сообщений получайте из файла конфигураций (предусмотрите трансформации для конфигураций).

8. Доработайте функционал клиентского приложения, предоставив игроку возможность выгрузки своих партий с возможностью их пошагового просмотра: предусмотрите кнопки undo/redo выполнения полуходов, возврата к началу партии, перехода к концу партии, перехода к произвольному месту в партии (при наличии элемента управления, отображающего полуходы в партии).

9. Реализуйте клиентское приложение, предназначенное для взаимодействия с сервисом из задания 2 (админка).

10. [опционально] Дальнейшие возможные улучшения:
    - кастомизация вариантов правил (русские, международные, канадские, турецкие, поддавки, столбовые, …);
    - игра на время (контроли времени на партию с добавлением времени на ход);
    - редактор доски;
    - тренажёры для решения задач;
    - <придумать своё> :)