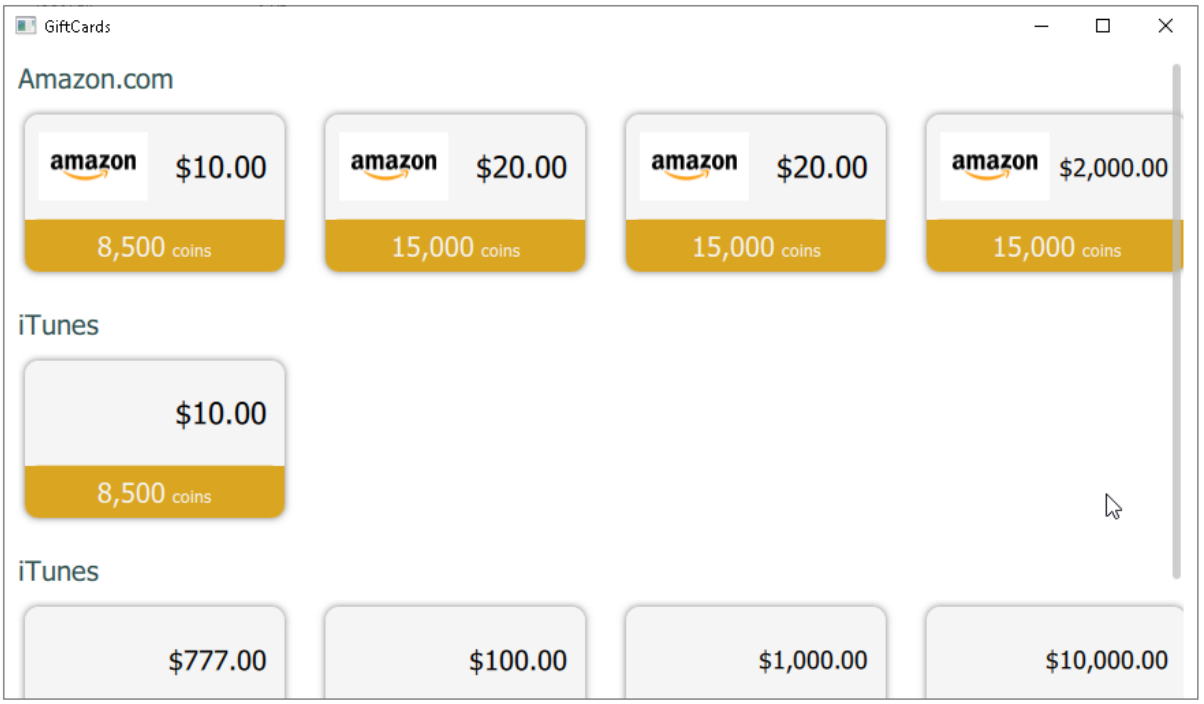
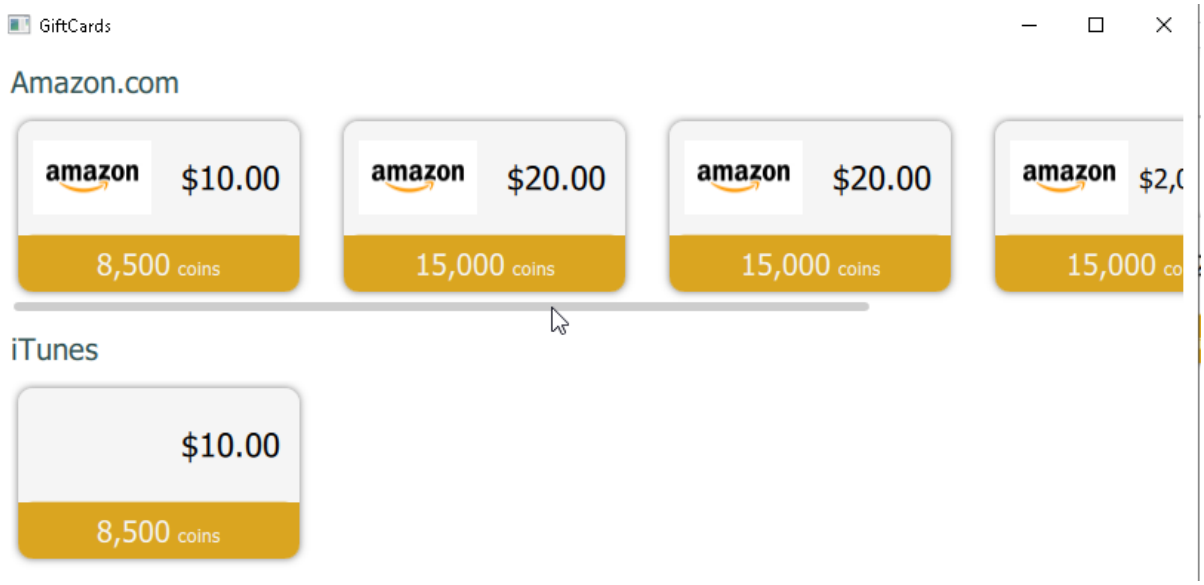


Приложении отображает подарочные карты разных провайдеров.

В приложении при необходимости есть вертикальный скролл.

Карты каждого провайдера располагаются в строке, в которой при необходимости есть горизонтальный скролл:



Общее описание

Карточки должны иметь форму как на скриншоте, с такими же тенями.

Приложении при запуске делает запрос по HTTP/HTTPs по адресу, который указан в конфиге приложения, если адрес в конфиге не указан, то используется адрес по умолчанию - http://office.vedisoft.ru/files/providers.json

Если запрос не успешен, то отображается диалог с ошибкой.
Во время запроса показываем спиннер. Находим способ его отобразить без модификации кода и произвольным адресом для json.

На карточках максимально должно отображаться 1000000.00$, размер шрифта для суммы зависит от величины (т.е. для 10$ шрифт больше, для 1000$ меньше).
То же самое для coins.

Провайдеры и карточки отображаются в порядке увеличения их id.
Картинки должны загружаться асинхронно - если получение картинки тормозит, то это не должно тормозить всю ленту - лента должна отобразиться, затем после загрузки картинки, она должна появиться в ленте, если получить картинку не удалось, то карточка будет без картинки.

При наведении на карту отображается tooltip с описанием (из поля "description" в json)

Цвет в нижней части карточки всегда одинаковый.

Не отображаем провайдеров без валидных карточек.

Выводим подробный лог с уровнями Debug, Info, Warning и Error.
Уровень лога задаётся в конфиг файле.
Конфиг файл должен быть в формате ini.

Покрываем парсинг json юнит тестами.

Формат json


```
{
   "providers": [
      {
         "id": 1,
         "title": "Amazon.com",
         "image_url":
"http://g-ec2.images-amazon.com/images/G/01/social/api-share/amazon_logo_500500.png",
         "gift_cards": [
            {
              "id": 1,
              "featured": false,
              "title": "$10 Amazon.com",
              "credits": 8500,
              "image_url":
"http://g-ec2.images-amazon.com/images/G/01/social/api-share/amazon_logo_500500.png",
              "codes_count": 101,
```

```
            "currency": "USD",
            "description": "Buy everything from Amazon. It's great.",
            "redeem_url": "http://www.amazon.com"
          },
...
```


Есть массив провайдеров, внутри каждого массив карточек.
Внизу карточки (coins) отображаем значение из "credits".
Поля "image_url" у провайдера и "featured", "codes_count", "currency", "redeem_url" у карточек не используются в UI, но должны быть сохранены в БД.
Значение суммы в карточке парсим из строки "title", предполагаем, что формат строки такой, что в начале может стоять некоторое число пробелов, затем знак $, далее сумма.

Карточка невалидна, если не удалось распарсить сумму, если нет "credits" или в нём не число. Или если любое из этих чисел больше миллиона.
Провайдер не валиден, если title пустой после trim.
Также провайдер или карточка невалидны, если такой id уже был обработан.

БД

Помимо отображения данные о карточках должны сохраняться в Sqlite БД.
Причём после получения данных из сети, они сначала записываются в БД, а затем через модель отображаются из БД в GUI.
Предыдущие данные удаляются при успешном получении новых.
Нужно предложить схему БД и реализовать после обсуждения.

Порядок выполнения задания

Важно выполнять задание в правильном порядке:

1. Внимательно прочитать задание, задать вопросы
2. Предложить общую архитектуру, обсудить, получить одобрение
3. Предложить формат конфиг файла и логов, обсудить, получить одобрение
4. Создать парсер json, покрыть его тестами
5. Создать UI, использовать простую модель получения данных из памяти, но с учётом дальнейшего чтения из БД
6. Предложить схему БД, обсудить, получить одобрение
7. Создать логику сохранения json данных в БД, покрыть тестами запись и чтение из БД
8. Создать логику получения данных по сети
9. Интегрировать логику получения данных по сети, сделать состояния в UI (загрузка в процессе, загрузка завершена успешно, ошибка), записывать полученные данные в БД
10. Переделать модель на чтение данных из БД
11. Привести приложение к итоговому виду, устранить все замечания по коду и логике
12. Протестировать приложение, написать тестовые сценарии