Тестовая задача.

# Описание

Нужно написать миниатюрную версию платёжного шлюза.

Компоненты:

* Сервер, имитирующий работу банка
* Демо-риложение, реализующее api банка(сервера)

По результатам выполнения тестового задания нужно предоставить:

* сервера и демо-приложения
* инструкции по запуску
* (опционально, но очень желательно) документ в котором будет описано какие архитектурные решения были приняты и почему

# Демо-приложение

Должно продемонстрировать работу API. Также нужно продемонстрировать работу при вводе неверных данных, неверной логике работы клиента. Можно сделать простое консольное приложение. Все данные задать в виде констант прямо в коде. **Важно**: не включайте в код данные настоящих банковских карт. Используйте только тестовые / фиктивные карты.

# Сервер (эмулятор процессинга банка)

Нужно написать сервер, имитирующий работу процессинга банка.

Общение между приложением и сервером происходит путём HTTP запросов и ответа в виде json(xml).

Нужно завести как минимум две тестовые карты. У каждой есть номер, срок действия, CVV, лимит средств. На одной карте бесконечный лимит средств, на второй ограниченная сумма. При выполнении запросов сервер должен проверять корректность переданных данных и доступный лимит средств.

Для уменьшения объёма работы данные можно хранить только в памяти, не сохраняя в БД или на диск.

Также не обязательно писать сервер с нуля, можно использовать какой-нибудь.

## API

### Оплата товара или услуги

Pay(order\_id, card\_number, expiry\_month, expiry\_year, cvv, cardholder\_name, amount\_kop)

order\_id - идентификатор заказа на стороне продавца. Должен быть уникальным.

amount\_kop - сумма в копейках,

card\_number - номер банковской карты

expiry\_month, expiry\_year - срок действия карты

cvv - верификационный код (3 цифры)

cardholder\_name - имя владельца карты (необязательно)

Возвращает ok или код ошибки. Ошибки могут быть разные (банк не смог обработать операцию, неверный номер карты и т.д.). Нужно будет завести коды ошибок, которые понадобятся / посчитаете нужными.

### Проверка статуса транзакции

GetStatus(order\_id)

order\_id - идентификатор заказа на стороне продавца

Возвращает статус платежа: сделан, не сделан, возвращён и т.д. Нужно будет добавить те статусы, которые понадобятся.

### Возврат средств

Refund(order\_id)

order\_id - идентификатор заказа на стороне продавца

Полный возврат платежа клиенту.

Возвращает результат операции: ok или код ошибки. Нужно будет завести коды ошибок, которые понадобятся / посчитаете нужными.

# Дополнительно

* В качестве языка разработки использовать .net/C#.
* (По желанию) Api реализовать в виде отдельной библиотеки, которую будет потом использовать демо-приложение.
* Покрыть тестами нужный по вашему усмотрению код. Для написания тестов использовать xUnit.   
  (По желанию) При написании тестов взять во внимание, что ваш сервер является боевым и отправка запросов на него, для выполнения определенных тестов, не желательна – замокать ответы от сервера.