DOI 10.37882/2223–2966.2022.11–2.25

РАЗРАБОТКА ЧАТ-БОТА ДЛЯ ИГРЫ В МОНОПОЛИЮ «В КОНТАКТЕ»

DEVELOPMENT OF A CHATBOT FOR THE MONOPOLY GAME VKONTAKTE

A. Pimanov

Summary. Chatbots are already an important part of the daily life of many services. For the financial sector, these are usually bots taking on the role of consultants, in social networks, bots carry entertainment and interactive functions.

This article discusses the implementation of a chatbot for the social network "In Contact", which organizes the Monopoly gameplay in group chats, through the use of the PHP scripting language and interaction with the VK API. Analytical work was carried out to identify the advantages and disadvantages of the implemented project, as well as the possibilities of its modification were evaluated.

This code can be integrated into other chatbot projects in order to expand their functionality, which can subsequently lead to attracting an active audience.

Keywords: chatbot, script, VK API, PHP, JSON, Monopoly, VKontakte.

Пиманов Андрей Евгеньевич
Технический институт (филиал) федерального государственного автономного образовательного учреждения высшего образования "Северо-Восточный федеральный университет имени М.К. Аммосова" в г. Нерюнгри

Аннотация. Чат-боты уже являются важной частью повседневной жизни многих сервисов. Для финансовой сферы — это как правило боты, берущие на себя роль консультантов, в социальных сетях боты несут развлекательную и интерактивную функции.

В данной статье рассмотрена реализация чат-бота для социальной сети «В Контакте», организующего игровой процесс Монополии в групповых чатах, посредством использования скриптового языка PHP и взаимодействия с VK API. Проведена аналитическая работа по выявлению преимуществ и недостатков реализованного проекта, а также оценены возможности его модификации.

Данный код возможно интегрировать в другие проекты чат-ботов с целью расширения их функционала, что впоследствии может привести к привлечению активной аудитории.

Ключевые слова: чат-бот, скрипт, VK API, PHP, JSON, Монополия, ВКонтакте.

Монополия — классическая настольная игра о торговле недвижимостью для двух и более человек. Цель игры — рационально используя стартовый капитал остаться единственным игроком, избежавшим банкротства.

Реализация уже существующей Монополии является абсолютно не тривиальной задачей, ведь данная игра обладает богатым набором механик и правил, которые не позволят при разработке избежать противоречащих друг другу участков кода. К тому же такие фрагменты являются многочисленными и нарушаемые между ними взаимоотношения зачастую запутаны. Усложняет задачу так же факт того, что итоговая реализация должна быть умещена в чат-боте. Вследствие данных факторов, по мере продвижения разработки необходимо расширять набор инструментов, включаемых в проект.

Так как целью проекта являлась разработка именно чат-бота, для реализации был выбран скриптовой язык общего назначения PHP. Благодаря чему взаимодействие с социальной сетью было максимально упрощено. Связь с «В Контакте» удалось реализовать с помощью VK API. VK API — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу.

В основе проекта заложен объектно-ориентированный подход. Все функции были сгруппированы по их назначению и перенесены в методы соответствующих классов. На рисунке 1 изображена схема исполняемых файлов проекта, отображающая взаимодействие между файлами.

Файлы config.php, vk.php и monopoly.php описывают классы, в то же время файл script.php является скриптом, в котором сосредоточена вся логика проекта. Класс Config содержит константы для взаимодействия с VK API, такие как версия API, токен сообщества и строку для подтверждения сервера. Класс VK описанный в файле vk.php содержит методы для взаимодействия с VK API. Данный класс использует константы из класса Config и содержит следующие методы: конструктор для подтверждения сервера и сбора информации из входящего JSON запроса; метод отправки ответа в формате JSON; метод для отправки сообщения; метод для загрузки изображений на сервера «В Контакте»; метод для получения url адреса по которому необходимо будет загрузить изображения; метод для сохранения изо-

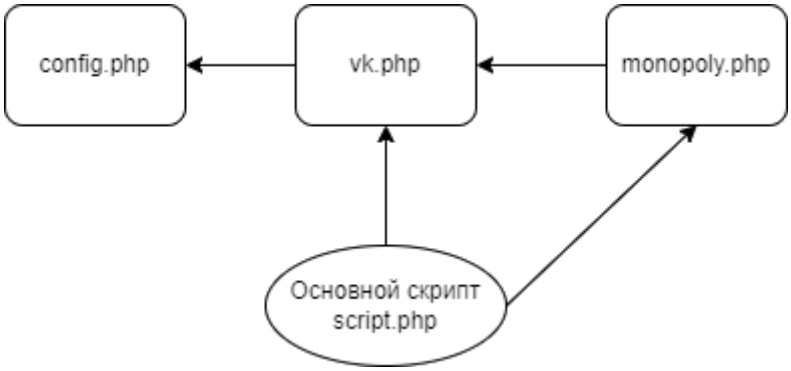

Рис. 1. Схема проекта

бражения на серверах «В Контакте». Класс Monopoly объединил в себе методы затрагивающие реализацию любых игровых механик, при этом его методы обращаются к методам класса VK. Класс Monopoly содержит следующие методы: прорисовка игрового стола; размещение элементов на игровом столе; размещение фишек на игровом столе; сбор баланса игроков; селектор цвета; определение текущего хода; передача хода; проверка нарушения игровых условий.

Скрипт script.php, взаимодействующий со всеми перечисленными классами посредственно и непосредственно, не способен реализовать эффективную работу с множеством игровых сессий, не прибегая к СУБД. По этой причине была создана база данных monopoly, содержащая следующие таблицы: vk_donut; sample_map; sample_game_parameters; information_about_fields.

Таблица vk_donut состоит из одного естественного первичного ключа целочисленного типа. Поле предназначено для хранения идентификаторов пользователей, которые смогут начать игру в Монополию. Такое ограничение позволяет сэкономить вычислительные ресурсы.

Таблица sample_map состоит из суррогатного первичного ключа целочисленного типа, столбца name типа varchar для названия игрового поля, целочисленных полей recruiter и level для хранения информации о владельце поля и количестве звёзд для каждой франшизы. Посредством копирования данной таблицы, получается новая таблица, хранящая информацию о всех игровых полях в сессии. Созданная таким образом таблица будет носить название {$peer_id}_map, где {$peer_id} идентификатор беседы.

Таблица sample_game_parameters состоит из суррогатного первичного ключа целочисленного типа, столбца parameters типа varchar для хранения названия параметров и целочисленного столбца value для значения параметров. Посредством копирования данной таблицы, получается новая таблица, хранящая информацию о игровой сессии и действующей в них правилах. Созданная таким образом таблица будет носить название {$peer_id}_game_parameters. Данная таблица включает следующие игровые параметры:

Таблица information_about_fields состоит из: суррогатного первичного ключа целочисленного типа; столбцов name и franchise типа varchar хранящих имена игровых полей и названий сегментов рынка; столбца rent типа JSON содержащего размеры арендных плат для полей в зависимости от количества построенных филиалов; так же из целочисленных столбцов field_cost, field_deposit, field_redemption и purchase_of_a_branch хранящих информацию о стоимости поля, вознаграждения за залог поля, размера выкупа поля и стоимости поля соответственно; столбцов coordinates_for_chips, coordinates_for_field_colors, coordinates_for_stars и coordinates_for_text типа JSON для хранения координат наложения фишек, цветов полей, звёзд и текста поверх заготовленного изображения. Данная таблица хранит неизменяемую в процессе информацию об игровых полях, которую нет необходимости дублировать в таблице sample_map и всех от неё производных.

При формировании игровой сессии создаётся ещё одна таблица. Своей структурой, она не повторяет уже существующую, а имеет свою независимую. Эта же таблица необходима для сохранения информации о каждом игроке. Она носит имя {$peer_id}_players и состоит из следующих столбцов: суррогатного первичного ключа целочисленного типа; целочисленных столбцов user_id и color для хранения идентификатора пользователя в социальной сети и номера цвета игрока соответственно; столбец money типа float для хранения денежной суммы доступной игроку; целочисленные столбцы cell, refused_the_auction, arrest для хранения информации о номере поля на котором стоит игрок, статусе

Таблица 1

Название параметра	Значение по умолчанию	Описание параметра
type_game	2	Определяет формат игры из следующих вариантов: 2 — два игрока; 3 — три игрока; 4 — четыре игрока; 5 — командная игра 2 на 2;
current_move	1	Определяет номер цвета игрока, которому принадлежит текущий ход
waiting_for_a_move	0	Выступает в роли ограничителя, не позволяющего выполнить команды не по месту их назначения
building_a_branch	1	Указывает на возможность строительства филиалов
field_participating_in_auction	0	Указывает на то, что поле разыгрывается на аукционе
cost_of_the_field_at_auction	0	Описывает текущую стоимость поля на аукционе
current_move_auction	0	Определяет номер цвета игрока, которому принадлежит текущий ход на аукционе
fine	0	Определяет размер текущего штрафа
duplicate	0	Указывает количество выброшенных дублей подряд
load_game	0	Указывает на то, что игра была сформирована и начата

участника аукциона и количестве ходов проведённых игроком в тюрьме соответственно; два столбца типа JSON — jackpot для хранения выбранных чисел игроком находящимся на поле Джекпот и столбец deal для хранения объектов сделки осуществляющейся между игроками.

На данном этапе для скрипта script.php описаны все классы и некоторые инструменты для начала работы. Сначала в скрипт подключается класс VK и создаётся его объект vk, следующим этапом инициализируется подключение к базе данных. Далее необходимо проверять тип пришедшего события, и если оно соответствует message_new тогда скрипт должен приступить к извлечению идентификатора беседы, автора сообщения и его содержимого. После подключается класс Monopoly и инициализируется его объект, присваиваются полям класса объекты vk и mysqli. Из таблицы vk_donut запрашивается идентификатор пользователя, совпадающий с идентификатором автора сообщения. Если идентификатор автора сообщения присутствует в таблице, он сможет начать игру и выбрать её режим. Пользователи, которые не входят в данный список, таким функционалом не владеют, но могут присоединиться к сформированной игре. После формирования игры в базе данных появятся 3 новых таблицы, описанные ранее и начинающиеся на {peer_id}. Это необходимо для того, чтобы ограничить диапазон пользователей способных начать игру, и в последующем обеспечить экономный расход вычислительных ресурсов.

Стоит учитывать, что проверка на наличие идентификатора автора в таблице vk_donut должна выполняться только в том случае, если в базе данных отсутствуют 3 таблицы, описывающие игровую сессию. Именно с помощью этой проверки отделяются команды для формирования игры и игровые команды. Перед выполнением игровых команд необходимо записать все игровые параметры из соответствующей таблицы в соответствующие переменные и проверить значение параметра load_game. Данный параметр позволяет выполнять игровые команды даже по мере выхода игроков из игры.

Теперь следуют блоки кода ответственные за выполнения игровых команд. Здесь используется разделение на команды доступные в любое время игры и команды доступные лишь во время проведения аукциона. Данная проверка возможна только благодаря таблице, хранящей параметры. В каждом таком блоке для определения введённой команды используется оператор switch. Подаваемое ему значение сравнивается с true, что даёт возможность занести в каждый case более сложное условие. Это позволяет более гибко использовать оператор switch, в отличии от стандартного варианта его эксплуатации. Почти каждая команда содержит обращение к методу checking_game_conditions класса Monopoly, с передачей ассоциативного двумерного массива, где ключом является условие для проверки, а значением является массив с индивидуальными проверяемыми аргументами для каждого ключа. Если хоть одно условие не выполняется, метод возвращает текстовое сообщение об ошибке, которое будет отправлено в беседу. В ином случае метод вернёт false, что приводит к выполнению основной функции команды. Например, команда “бросить кубики” устроена следующим образом. Коман-

да с помощью функции rand генерирует два значения на кубиках, сохраняя их в массив $dice из двух элементов. Выбрав случайные значения, происходит проверка поля таблицы arrest для игрока, бросившего кубики. Если игрок, находящийся в тюрьме, выбросит дубль, то значение arrest обнулится, и игрок сможет выбросить кубик повторно, чтобы продолжить передвижения по игровым полям. В ином случае игрока либо заставят заплатить залог для выхода из тюрьмы, либо игрок сможет попробовать выбросить дубль во время своего следующего хода, при том условии что таких попыток не было совершено на момент броска три. В случае, когда игроку не удаётся выйти из тюрьмы и количество попыток при этом не превышает трёх, происходит обращение к методу pass_the_move класса Monopoly ответственному за передачу хода следующему игроку. Очень часто каждая выполняемая команда приводит к отправке особенного сообщения в беседу с помощью метода assembling_a_game_card опять же принадлежащего классу Monopoly. Данный метод отличается тем, что формирует изображение игрового стола и прикрепляет его к отправляемому сообщению. Устройство метода assembling_a_game_card рассмотрено ниже. Если игрок не находится сейчас в тюрьме, на выполнение последует другой фрагмент кода. Посредством прибавления значения номера ячейки, на которой располагается фишка игрока, к сумме элементов массива $dice, достигается передвижение фишки игрока по полю. Также фиксируется момент, когда игрок проходит очередной круг для начисления соответствующих бонусов. Вследствие данной фиксации изменяются таблицы {$peer_id}_game_parameters и {$peer_id}_players таким образом, чтобы игрок выбросивший дубль смог совершить ход ещё раз или попасть в тюрьму за выброс дубля более трёх раз подряд. После того как фишка игрока переместилась по игровому полю, следует провести работу с тем полем на которое игрок попал. Для поля "Старт", с которого начинается передвижение игроков, начисляется дополнительный бонус за прохождения круга в размере 1000 денежных единиц. Для полей "Подоходный налог" и "Налог на роскошь", взимаются соответствующие денежные штрафы. Для поля "Полицейский участок" отправляется сообщение об экскурсии по полицейскому участку. Поле "Джекпот" представляет из себя большой алгоритм для реализации ставок на выбранные числа, выпавшие на кубике. Поле "Полицейский" отправляет игрока в тюрьму. Поле "Шанс" вызывает одно из случайных событий: телепортация в случайное место на карте; награждение или штраф в разном размере и так далее. Другие исходы касаются полей с предприятиями, которые могут быть в хозяйстве у одного из игроков, заложены или вовсе никому не принадлежать. Если игрок сделавший шаг наступил на поле другого игрока, необходимо предложить игроку бросившему кубики оплатить штраф. Если поле в залоге, стоит просто передать ход следующему игроку. Для поля не имеющего владельца, игроку, бросавшему кубики, необходимо предложить купить поле или выставить его на аукцион. Взаимосвязь между командами является достаточно высокой. В процессе меняется значение параметра waiting_for_a_move, который вместе с ранее рассмотренным методом проверки игровых условий checking_game_conditions не позволяет нарушать ход игры.

Выше упоминался метод assembling_a_game_card, являющийся частью класса Monopoly, используемый для формирования и прикрепления изображения игрового стола. Данный метод вызывается очень часто и обеспечивает игрокам интерактивное сопровождение их игровой сессии. Итоговое изображение формируется из наложения друг на друга заранее заготовленных изображений и текста. Первым слоем всегда располагается игровая карта без изображений логотипов компаний. На ней изображена лишь общая разметка с игровыми полями, такими как: поле "Старт"; "Полицейский участок", поле "Джекпот" и так далее. Следующим шагом возле соответствующих предприятий наносится изменяемая в зависимости от их статуса информация о стоимости и аренде. Дальше наносится фоновый цвет поля, чтобы указать принадлежность поля игроку по его цвету. Теперь можно наложить логотип поля. Следующим этапом накладываются фишки игроков, далее звёзды и индикатор статуса заложенности поля. Такой порядок необходим, чтобы избежать перекрытия элементов друг другом. Изображение сохраняется под названием {$peer_id}_map в расширении jpg. Работа с графикой осуществляется с помощью встроенной библиотеки GD library. После, изображение загружается на сервера «В Контакте», там же сохраняется и отправляется в групповой чат.

Для загрузки изображений в классе VK предусмотрены следующие три метода: photos_get_messages_upload_server, photos_upload_server и photos_save_messages_photo. Первый из них необходим для получения url адреса, по которому будет загружено изображение. Второй используется для загрузки изображения по ранее полученному адресу с помощью cURL. И третий для сохранения изображений на серверах «В Контакте» с помощью данных, полученных от предыдущего метода. После сохранения изображения, отправить сообщение можно с помощью метода messages_send принимающего аргументы: peer_id — идентификатор группового чата; message — текст сообщения; attachment — идентификатор вложения; keyboard — клавиатура в формате JSON используемая в проекте для обеспечения лучшего игрового опыта.

В данном материале была рассмотрена реализация игрового чат-бота для игры в Монополию в социальной

сети «В Контакте». Преимуществами данного алгоритма можно считать грамотное использование расширений и библиотек, а также соответствие стандарту оформления кода PSR. При наличии специальных навыков по работе с PHP, SQL и базами данных, модификация скрипта не должна составить слишком больших затруднений, хотя и лёгкой данную процедуру считать нельзя. Например, без больших усилий можно расширить максимальное количество игроков в рамках игровой сессии или внести изменения позволяющие создавать в рамках одной беседы несколько игровых сессий одновременно. С кодом проекта можно ознакомиться на GitHub.

ЛИТЕРАТУРА

1. Официальный веб-сайт «В Контакте», раздел «Для разработчиков» [Электронный ресурс]. URL: https://dev.vk.com/reference (дата обращения: 04.02.2022).
2. Официальный веб-сайт «php.net», раздел «Руководство по PHP» [Электронный ресурс]. URL: https://www.php.net/manual/ru/index.php (дата обращения: 04.02.2022).
3. Карпова И.П. Базы данных. — М.: Питер, 2013. С. 3–31
4. Кузнецов М.В., Симдянов И.В. Самоучитель PHP 7. — СПб.: БХВ-Петербург, 2018. С. 143–151.
5. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL. Полное руководство. — Вильямс, 2018. С. 95–148.
6. Официальный веб-сайт «GitHub», репозиторий [Электронный ресурс]. URL: https://github.com/Aiciokizava/bot-for-monopoly

Журнал «Современная наука: актуальные проблемы теории и практики»

Политехнический институт (филиал) СВФУ им. М.К. Аммосова