Клиентская часть ПС была реализована посредством APIWindowsForms. Выбор обоснован тем, что данный интерфейс является частью Microsoft .NETFramework и его использование упрощает доступ к элементам MicrosoftWindows. Также, данный API достаточно просто связать с СУБД на базе MySQLserver [1].

Клиент состоит из логики отправки запросов к серверному приложению и предоставляет следующие возможности: 1) регистрация и аутентификация пользователя в системе; 2) возможность просмотра пройденных тестовых заданий; 3) возможность сравнения результатов по группе пользователей/студентов; 4) возможность сравнения результатов по нескольким группам пользователей/студентов.

Список использованных источников:
1.Wikipedia [Электронный ресурс]. – Режим доступа: http://wikipedia.org – Дата доступа: 15.03.2018.

РАЗВЁРТЫВАНИЕ МИКРОСЕРВИСОВ

Институт Информационных технологий БГУИР,
г. Минск, Республика Беларусь

Петрович А.С.

Образцова О.Н. – доцент каф. ИСиТ, , к.т.н., доцент
Бакунова О.М. – ст. преподаватель каф. ИСиТ, м.т.н.
Бакунов А.М. - ст. преподаватель каф. ИСиТ, м.т.н.
Калитеня И.Л. - ассистент каф. ИСиТ, м.т.н.

В работе проведен анализ использования микросервисов в программировании и особенности использования микросервисных архитектур.

Микросервисы – это маленькие независимые сервисы, которые совместно работают для достижения общих целей. Этот концепт не нов – он был придуман более десяти лет назад, но популярность и широкое распространение приобрёл в недалёком прошлом. Это произошло ввиду быстрого роста IT сферы, которое создало таких гигантов индустрии как Гугл, Макрософт, Нетфликс, Амазон и прочих. И эти гиганты осознали, что монолитные системны крайне невыгодны ввиду проблем с их масштабированием. Чтобы разрешить эту проблему была создана сервис-ориентированная архитектура. Но сегодня и её уже недостаточно – на смену SOA приходит микросервисная архитектура.

Микросервисы привносят не так много улучшений по сравнению с сервисами, потому что MSA не является новшеством само по себе. Тем не менее, каноническое их представление имеет множество преимуществ даже в сравнении с родительской архитектурой. Но достичь этого можно только чётко следуя принципам построения MSA (рисунок 1).

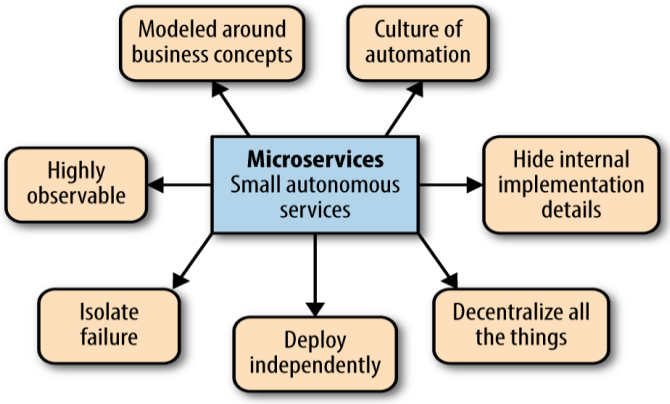

Рисунок 1 - Принципы MCA

Отказ от ответственности: микросервисы – это, сравнительно, очень молодая технология, так что некоторые не указанные на схеме принципы могут быть лучше в тех или иных случаях, также как представленные на рисунке ниже могут представлять собой проблему при развёртывании. Принципы построения должны определяться только архитектором системы.

Моделирование вокруг бизнес-процессов. Отличным способом разбить монолит на микросервисы – это смоделировать их вокруг бизнес-процессов (рисунок 2). В настоящих компаниях разные департаменты имеют слабые связи между собой, так что создание микросервисов, повторяющих их структуру может быть крайне полезным.

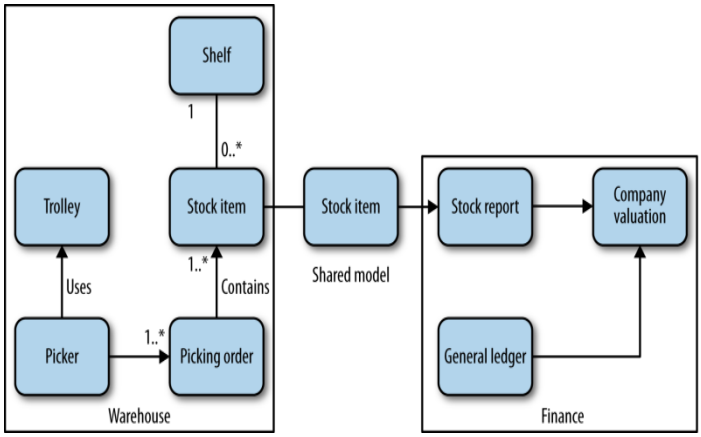

Рисунок 2 - Пример моделирования системы вокруг бизнес-процессов

Тем не менее, в случае создания микросервиса с нуля (вместо разбиения монолита) куда чаще можно столкнуться с проблемой неверного определения границ серциса из-за недостаточного понимания процессов на предприятии. На самом деле это – не большая проблема, потому что размер микросервисов подразумевает, что каждая ошибка может и должна быть исправлена быстро.

Сокрытие деталей внутренней реализации. Как показано на втором рисунке микросервисы, которые представляют собой складской и финансовый отделы “находятся в коробке”. Это означает, что детали реализации каждого сервиса должны быть сокрыты от прочих сервисов. В идеале микросервисы должны представляться другим сервисами только посредством API.

Децентрализация. Тенденции децентрализации наблюдаются во множестве современных программ (например, в Blockchain). Так что микросервисы тоже стремятся уйти от единой точки отказа. Это охначает, что какой-либо сбой одного компоненита не должен привести к краху целой системы. Это также означает, что при построении микросервисов мы стремимся избежать расположения их на единой платформе (золотое правило: один хост – один сервис) или даже в едином дата-центре в целях предотвращения потери работоспособности системы целиком ввиду отказа каких-либо физических серверов.

Изоляция сбоев. Потеря дата-центра не единственная проблемаL,с которой можно столкнуться при разработке систем, основанных на микросервисной архитектуре. Цель хорошего архитектора построить стабильную систему микросервисов. Когда одна из частей системы работает плохо это не должно повлиять на другие её части. Это – одно из ключевых преимуществ микросервисной архитектуры, потому что изоляция сбоев практически невозможна в монолитных системах.

Независимое развёртывание. Когда микросервисы разделены, децентрализованы и изолированы, приятным дополнением может стать возможность независимого их развёртывания. Это привносит множество преимуществ, но также налагает обязанности по предоставлению устаревшего API совместно с его новой версией потому что клиенты разрабатываемого той или иной командой микросервиса могут быть не готовы к переходу.

Микросервисы подразумевают, что разработчик может столкнётся с множеством ошибок. Но в отличие от монолитных систем, к ошибкам в MSA система готова. Как описано выше, все сбои должны быть изолированы, но для команд, участвующих в разработке и поддержке MSA важно быстро распознавать и реагировать на их появление. Для достижения данной цели важно иметь правильно сконфигурированные системы мониторинга.

Следует отметить, что хорошие микросервисные системы должны быть максимально автоматизированы. Чем больше сервисов реализуется, тем сложнее отследить их работу. Представленный программный комплекс способен выполнять базовую работу по проверке системы автономно. В идеале разработчик должны создать полностью автоматического наблюдателя, который контролирует выпуск новой версии сервиса в стадию производства и контролирует его в целях уменьшения затрат на его разработку.

Список использованных источников:

1. Микросервисы. [Электронный ресурс]. – Режим доступа: http://ra07.twirpx.net/1645/1645210_78383899/ newman_sam_ building_microservices.pdf'. – Дата доступа: 04.03.2018.

2. Микросервисы (microservices). [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/249183/. – Дата доступа: 04.03.2018.

3.Микросервисы. [Электронный ресурс].- Код доступа: https://www.youtube.com/watch?v=hqnilyrf81a. – Дата доступа: 15.03.2018.