МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

РЕФЕРАТ

Тема: Описание предметной области

Студентка гр. 4304 Лапцевич Д.А.

Санкт-Петербург
2019

СОДЕРЖАНИЕ

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Актуальность в настоящее время

С развитием мировых технологий экспоненциально увеличивается выбор девайсов с различной производительностью. Для того, чтобы веб приложение обеспечивало максимальный приток клиентов, а существующие клиенты не меняли выбор приложения в пользу конкурентного приложения, необходимо создавать производительные приложения, которые работают быстро и качественно на большинстве устройств, в том числе и с небольшой производительностью.

1.2. Архитектура клиент-серверного взаимодействия

Веб-приложения представляют собой тип программ, построенных по архитектуре “клиент-сервер”. Клиент-серверная модель представляет собой структуру приложения, которая распределяет задачи и нагрузки между поставщиками ресурсов и услуг, называемым серверами и теми, кто запрашивает эти услуги, называемыми клиентами. По сути клиенты и серверы представляют собой программное обеспечение. Как правило они расположены на различных вычислительных машинах, и обмениваются данными по вычислительной сети посредством сетевых протоколов, но иногда клиент и сервер могут находиться на одном компьютере. Хост сервера запускает одну или несколько серверных программ, которые распределяют свои ресурсы между клиентами. Клиент запрашивает содержимое сервера, но не передает ничего. Серверы ожидают запросы, а клиенты инициируют сеансы связи с ними.

Запросы клиента обрабатываются на сервере - там, где расположена База Данных и Система Управления Базой Данных (СУБД). Это дает преимущество в отсутствии пересылки больших объемов данных, а также запрос оптимизируется таким образом, что на него тратится минимум времени. Все это

повышает быстродействие системы и снижает время ожидания результата запроса. При выполнении запросов сервером существенно повышается степень безопасности данных, поскольку правила целостности данных определяются в базе данных на сервере и являются едиными для всех приложений, использующих эту БД [1].

Функции клиента:

- инициализация запроса серверу;
- обработка результатов запросов, полученных от сервера;
- представление результатов запроса пользователю в форме интерфейса пользователя.

Функции сервера

- прием запросов от клиента;
- обработка запросов;
- выполнение запросов к Базе Данных и их оптимизации;
- отправка результатов запросов клиенту;
- обеспечение системы безопасности;
- обеспечение стабильности многопользовательского режима работы.

1.3. Тестирование производительности

Тестирование производительности - это форма тестирования программного обеспечения, в котором основное внимание уделяется тому, как система работает под определенной нагрузкой. Тестирование производительности должно дать организациям диагностическую информацию, необходимую им для выявления и устранения узких мест.

Тестирование производительности лучше производить в процессе разработки приложения, так как есть вероятность обнаружить сбои, вызванные архитектурными решениями.

Для измерения производительности приложения проводятся нагрузочное тестирование, стресс-тестирование и тестирование на выносливость.

1.3.1. Нагрузочное тестирование

В тестовой среде можно выбрать нагрузку для приложения, например, одновременное использование приложения тысячью пользователями при выполнении обычных операций и измерение поведения приложения. Такое тестирование не будет проводиться созданием реальных пользователей. Для этого создаете программное обеспечение, которое помогает имитировать нагрузку.

1.3.2. Стресс-тестирование

При стресс-тестировании приложению создаются неблагоприятные условия, чтобы увидеть, как он себя ведет в этих ситуациях. Это позволяет понять, в какой момент оно сломается и какие узкие места у приложения имеются. Каждое приложение имеет точку прерывания, поэтому после отказа на стресс-тесте можно вносить изменения, зная что искать и когда ожидать проблем, и сделать все возможное, чтобы в тот момент, когда приложение терпело неудачу, оно делало это изящно и разумно.

1.3.3. Тестирование на выносливость

При тестировании на выносливость применяется нагрузка на определенное время. Так же, как необходимо знать, как ведет себя приложение с большим количеством пользователей, необходимо знать, как он будет вести себя в течение нескольких недель или месяцев. И лучше это узнать в тестовой среде, чем в режиме реального времени [2, 3].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. О модели взаимодействия клиент-сервер простыми словами. Арзитектура клиент-сервер с примерами. URL: https://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostymi-slovami-arxitektura-klient-server-s-primerami.html (дата обращения: 25.11.2019)
2. Fundamentals of Web Application Performance Testing. URL: https://msdn.microsoft.com/en-us/library/bb924356.aspx (дата обращения: 26.12.2019).
3. Fundamentals of Web Application Performance Testing. URL: https://stackify.com/fundamentals-web-applicationperformance-testing (дата обращения: 12.12.2019)