Исследование архитектурного решения

TO BE

Для разрабатываемой системы были приняты решения приведенные ниже.

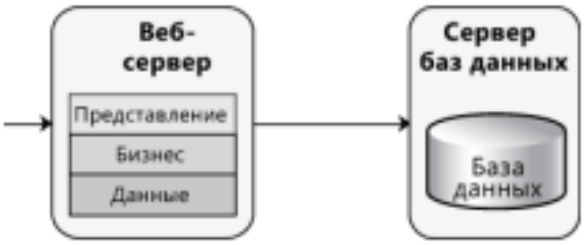
1. Тип приложения

Веб-приложения для выполнения преимущественно на сервере в сценариях с постоянным подключением.

Логика на стороне сервера – основа веб-приложения. Приложение осуществляет доступ к хранилищу данных на удаленном сервере базы данных.

2. Стратегия развёртывания

В процессе моделирования системы была выбрана стратегия нераспределённого развёртывания. При таком типе развертывания вся функциональность располагается на одном сервере, как показано на рисунке ниже.



Нераспределённое развёртывание.

Преимуществом данного подхода является простота и минимальные требования по количеству необходимых физических серверов. Также обеспечивается наилучшая производительность, поскольку взаимодействие между слоями осуществляется без пересечения физических границ между серверами.

Недостатком данной стратегии можно отнести совместное использование ресурсов всеми слоями приложения. Если один из слоев начинает слишком активно потреблять ресурсы, это негативно сказывается на работе всех остальных слоев. Кроме того, на используемые серверы накладывается требование в чёткой конфигурации.

2. Показатели качества

Показатели качества – это общие факторы, оказывающие влияние на поведение во время выполнения, дизайн системы и взаимодействие с пользователем. Следующим качествам уделяется внимание при разработке приложения:

- Безопасность (способность системы не допускать разглашение и утрату данных),
- централизованный доступ к данным,
- простота обслуживания (способность системы изменяться),
- надёжность (работоспособность системы в промежутке времени),
- производительность,
- удобство и простота использования,
- тестируемость (простые критерии проверки системы).

3. Пути реализации сквозной функциональности

С точки зрения обеспечения безопасности и надежности приложения реализуется эффективная стратегия аутентификации и авторизации. Проектирование эффективной стратегии управления исключениями важно для обеспечения надежности приложения. Приложение не остаётся в нестабильном состоянии после сбоя, и исключения не приводят к разглашению конфиденциальных данных.

В директории приведены диаграммы компонентов и развёртывания.

AS IS

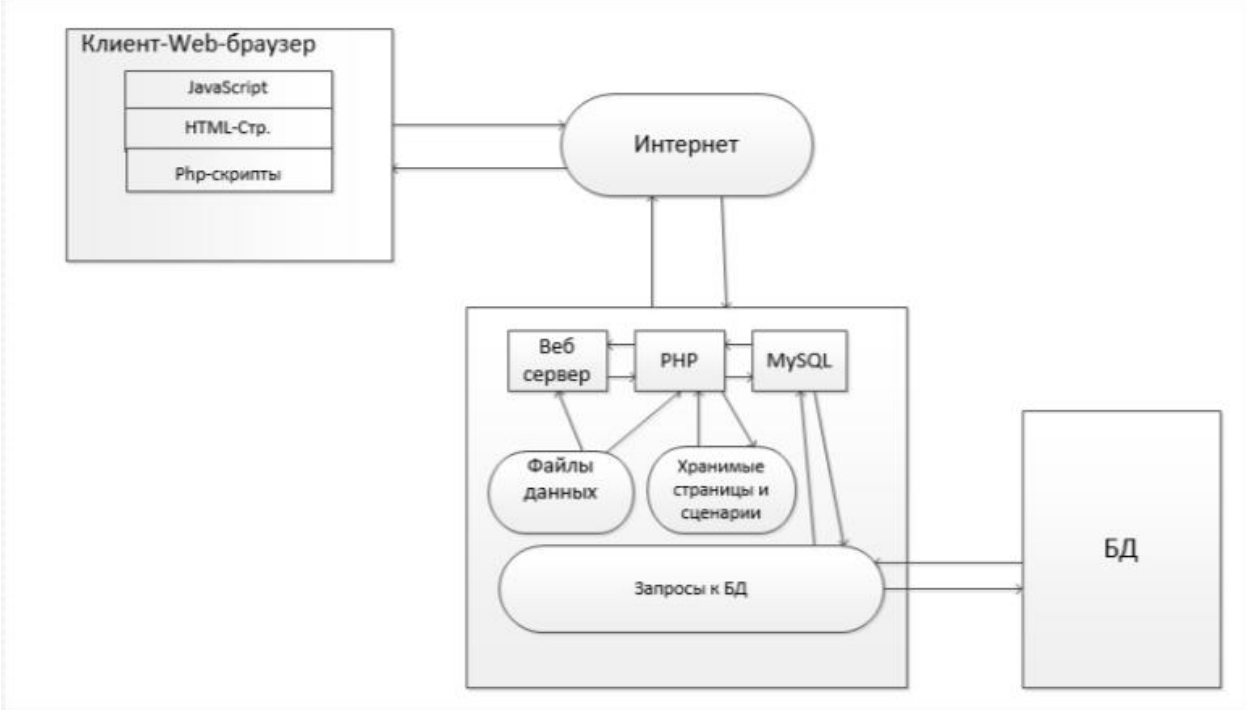
1. Архитектура приложения

В процессе разработки приложения наша команда придерживается клиент-серверной парадигмы. Простейшая форма системы клиент/сервер, называемая 2-уровневой архитектурой – это серверное приложение, к которому напрямую обращаются множество клиентов. Клиент/сервер представляет собой настольное приложение с графическим UI, обменивающееся данными с сервером базы данных, на котором в форме хранимых процедур располагается основная часть бизнес-логики.

Учитывается стратегия нераспределённого развёртывания и реализовывать основной функционал на серверной машине. Особое внимание уделяется критериям качества.

2. Обобщённое представление архитектуры

Структура приложения совпадает с запланированной и прослеживается на диаграммах компонентов и развёртывания в текущей директории. На рисунке представлена типовая структура веб-приложения.



Наша структура веб-приложения.

3. Диаграмма классов

Диаграмма классов приведена в директории.

Сравнение и рефакторинг

Архитектуры “As is” и “To be” имеют отличия. Архитектура “To be” более приближённая к клиент-серверному архитектурному стилю. Недостатки архитектуры “As is” можно объяснить ранней стадией разработки системы. Для того чтобы улучшить архитектуру приложения необходимо проведение рефакторинга.

В процессе рефакторинга необходимо помнить об основных принципах проектирования, таких как:

- Разделение функций (разделение приложения на отдельные компоненты с минимальным перекрытием функциональности),
- Принцип минимального знания (компонентам не известны внутренние детали других компонентов),
- Избегать повтора кода (определённая функциональность должна быть реализована в одной компоненте и не дублироваться в другом),
- Применить определённый стиль написания кода и присваивания имён для разработки (для получения единообразной модели).