Проектирование программного продукта

Архитектура программного обеспечения

В информационной системе главной составляющей является база данных.

По местоположению базы данных (далее – БД) определяют архитектуру базы данных. Все базы данных разделяются на локальные и удаленные.

Архитектура — это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы. Для выполнения операций с локальными БД разрабатываются и используются так называемые локальные приложения, а для операций с удаленными БД – клиент-серверные приложения.

В концепции «клиент-серверные» участвуют две стороны: клиент и сервер. Клиент – это заказчик той или иной услуги, а сервер – поставщик услуг. Клиент и сервер физически представляют собой программы, например, типичным клиентом является браузер. В качестве сервера можно привести следующие примеры: все HTTP сервера (в частности, Apache), MySQL сервер, локальный веб-сервер AMPPS или готовая сборка Denwer. Архитектура клиент-сервер определяет лишь общие принципы взаимодействия между компьютерами, детали взаимодействия определяют различные протоколы. Данная концепция нам говорит, что нужно разделять машины в сети на клиентские, которым всегда что-то надо и на серверные, которые дают то, что надо. При этом взаимодействие всегда начинает клиент, а правила, по которым происходит взаимодействие описывает протокол.

Существует два вида архитектуры взаимодействия клиент-сервер: первый получил название двухзвенная архитектура клиент-серверного взаимодействия, второй – многоуровневая архитектура клиент-сервер (иногда его называют трехуровневая архитектура или трехзвенная архитектура, но это частный случай).

Принцип работы двухуровневой архитектуры взаимодействия клиент-сервер заключается в том, что обработка запроса происходит на одной машине без использования сторонних ресурсов. Двухзвенная архитектура предъявляет жесткие требования к производительности сервера, но в тоже время является очень надежной.

В ходе исследования предметной области была выбрана клиент-серверная архитектура. Так как приложение общается с удаленным сервером и получает от него ответы на свои запросы.



Преимуществом модели взаимодействия клиент-сервер является то, что программный код клиентского приложения и серверного разделен. Если мы говорим про локальные компьютерные сети, то к преимуществам архитектуры клиент-сервер можно отнести пониженные требования к машинам клиентов, так как большая часть вычислительных операций будет производиться на сервере, а также архитектура клиент-сервер довольно гибкая и позволяет администратору сделать локальную сеть более защищенной.

К недостаткам модели взаимодействия клиент-сервер можно отнести то, что стоимость серверного оборудования значительно выше клиентского. Сервер должен обслуживать специально обученный и подготовленный человек. Если в локальной сети ложится сервер, то и клиенты не смогут работать (в качестве частного случая можно привести пример: мощности сервера не всегда хватает, чтобы удовлетворит запросы клиентов: время ожидания ответа от сервера может быть очень большим).

В качестве заключения нам стоит явно акцентировать внимание на том, что архитектура клиент-сервер не делит машины на только клиент или только сервер, а скорее позволяет распределить нагрузку и разделить функционал между клиентской частью и серверной.

Функциональное проектирование

Проектирование информационной системы происходит при помощи CASE средств, которые позволяют за короткий срок создавать схемы и реляционные модели программ.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

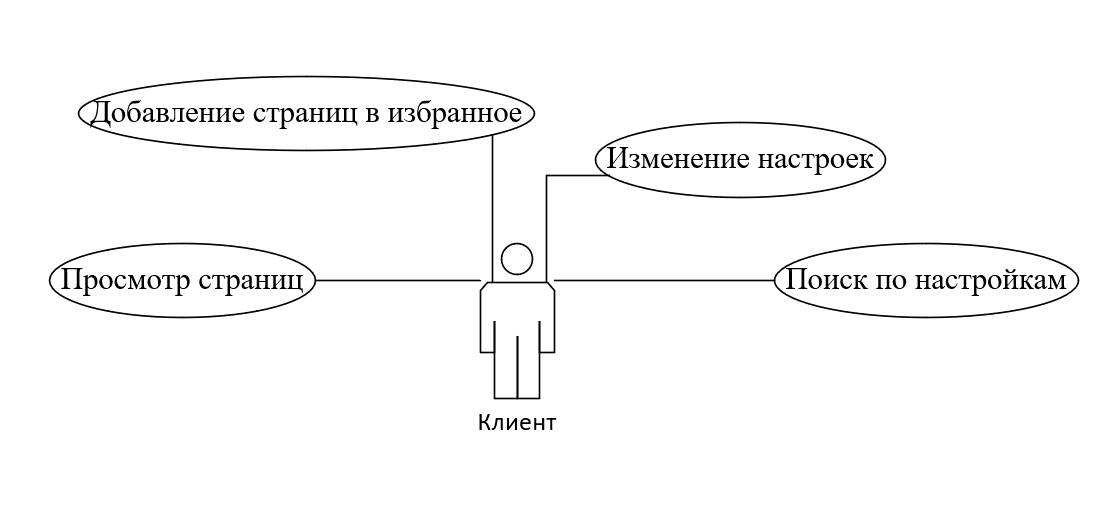
На рисунке 6 изображена диаграмма сценария использования, которая отображает действия, выполняемые сотрудником. «Клиент» является актёром. «Просмотр страниц», «Добавление страниц в избранное», «Изменение настроек», «Поиск по настройкам» - действия клиента. 

Рисунок 6 - Диаграмма вариантов использования

Контекстная диаграмма — (диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т. е. определяются функции, из которых состоит основная.

На контекстной диаграмме показаны входные данные, управление, механизм, выходные данные и функция (рисунок 7).

В центре диаграммы показана функция «Работа приложения Summary». Левая стрелка - входные данные «Запрос от клиента». Верхняя стрелка - управление «Протоколы передачи и обработки данных». Нижняя стрелка – механизмом является сотрудник отделения «Удаленный сервер». Правая стрелка - выходные данные «Ответ пользователю».

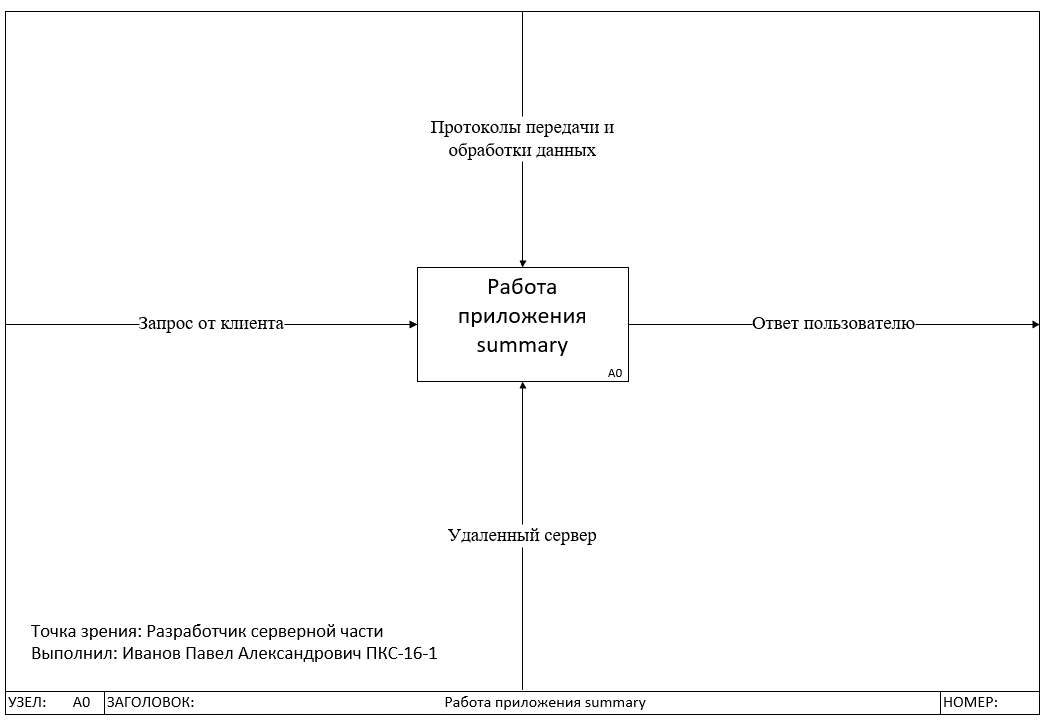


Рисунок 7 - Контекстная диаграмма

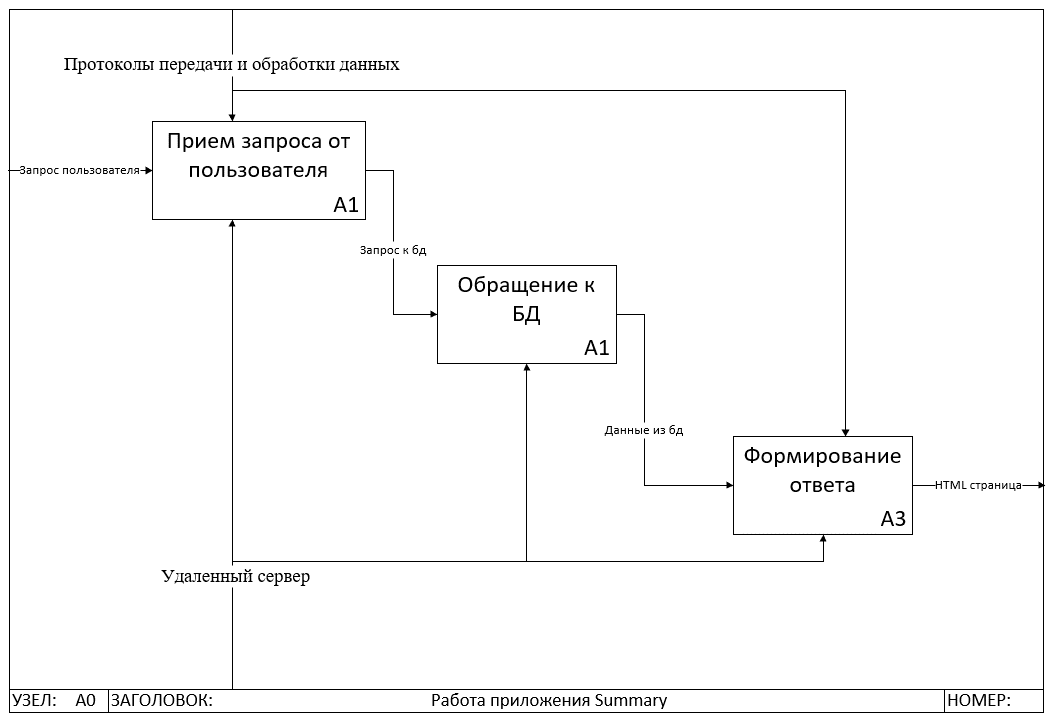


Рисунок 8 – Диаграмма декомпозиции

Диаграмма декомпозиции – предназначена для детализации работы. На рисунке 8 показана диаграмма декомпозиций, которая расписывает функцию. На этой диаграмме показаны 4 функции. «Прием запроса от пользователя» – Сервер принимает Http запрос пользователя и приступает к обработке. После приема запроса сервер приступает к его обработке и смотря по какому протоколу был сделан запрос выполняет соответствующую функцию - «Обращение к БД». Далее формируется ответ пользователю в функции «Формирование ответа», которая в последствии и возвращается клиенту на его устройство.

Проектирование базы данных

Прежде чем приступить разработке программного обеспечения необходимо спроектировать базу данных, а именно, определить с какими данными будут работать участники системы, и чем данные связаны между собой. В этом заключается процесс проектирования.

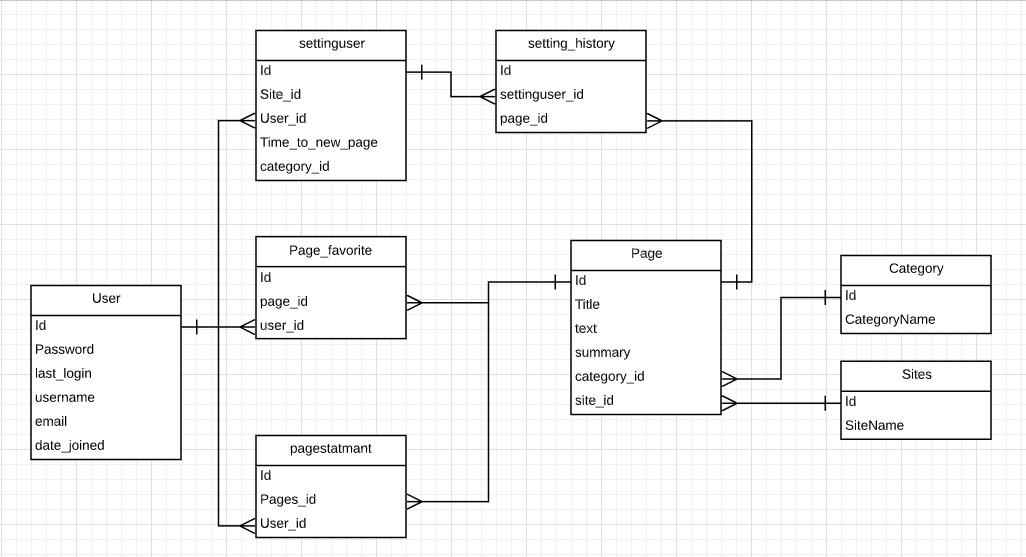
Проектирование базы данных не ограничивается только этим этапом разработки, а проходит на протяжении всей разработки, до того момента пока в системе не появятся данные, которые нельзя потерять. Результатом проектирования базы данных является ER-модель.

На рисунке 9 представлена разработанная ER-модель, которая демонстрирует сущности, и связи между таблицами.

Представлены 7 таблиц:

1. Category
2. Sites
3. Page
4. settingUser
5. PageStatmant
6. Page\_favorite
7. User
8. Setting\_history

Таблицы связаны при помощи связи «один ко многим» поле id является первичным ключом.



|  |  |  |
| --- | --- | --- |
| Category | | |
| Поле | Тип данных | Описание |
| Id | integer | Уникальный идентификатор |
| CategoryName | Varchar(100) | Название категории |
| Sites | | |
| Поле | Тип данных | Описание |
| Id | integer | Уникальный идентификатор |
| SiteName | Varchar(100) | Название сайта |
| Page | | |
| Id | integer | Уникальный идентификатор |
| Title | Varchar(100) | Заголовок статьи |
| Text | Text | Тело статьи |
| Summary | Varchar(150) | Краткое описание статьи |
| Category\_id | integer | Ссылка на категорию |
| Site\_id | integer | Ссылка на сайт |
| setting\_history | | |
| Id | integer | Уникальный идентификатор |
| Settinguser\_id | integer | Ссылка на таблицу конфигурации юзера |
| Page\_id | integer | Ссылка на сайт |
| settinguser | | |
| Id | integer | Уникальный идентификатор |
| Site\_id | integer | Ссылка на сайт |
| User\_id | integer | Ссылка на пользователя |
| Time\_to\_new\_page | Enum(‘1’,’2’,’6’,’24’) | Время через сколько присылать статьи |
| Category\_id | integer | Сслыка на категорию |
| Page\_favorite | | |
| Id | integer | Уникальный идентификатор |
| Page\_id | integer | Ссылка на статью |
| User\_id | integer | Ссылка на пользователя |
| Pagestatmant | | |
| Id | integer | Уникальный идентификатор |
| Page\_id | integer | Ссылка на статью |
| User\_id | integer | Ссылка на пользователя |
| User | | |
| Id | integer | Уникальный идентификатор |
| Password | Varchar(200) | Пароль |
| Last\_login | DateTime | Дата последнего входа |
| username | Varchar(100) | Логин |
| email | Varchar(100) | Электронная почта |
| Date\_joined | DateTime | Дата регистрации |

Представленная база данных имеет третью нормальную форму так, как она соответствует первой нормальной форме (каждый атрибут картежа может содержать только одно значение) и второй нормальной форме (каждый не ключевой атрибут зависит от первичного ключа), а также не содержит транзитивных и функциональных зависимостей неключевых атрибутов от ключевых (все независимые части вынесены в отдельные таблицы).