Столыпинский вестник

Научная статья

Original article

УДК 004

АРХИТЕКТУРА ПРИЛОЖЕНИЙ. ЧЕМ ЯВЛЯЕТСЯ? ДЛЯ ЧЕГО ИСПОЛЬЗУЕТСЯ? ОСНОВНЫЕ ВИДЫ И КРИТЕРИИ ХОРОШЕЙ АРХИТЕКТУРЫ

ARCHITECTURE OF APPLICATIONS. WHAT IS? WHAT IS IT USED FOR? MAIN TYPES AND CRITERIA OF GOOD ARCHITECTURE

Бежик А.А., Студент бакалавриата 3 курс, МИРЭА - Российский технологический университет (РТУ МИРЭА)

Мажей Я.В., Студент бакалавриата 3 курс, МИРЭА - Российский технологический университет (РТУ МИРЭА)

Bezhik A.A., 3rd year undergraduate student, MIREA - Russian Technological University (RTU MIREA)

Mazhey Ya.V., 3rd year undergraduate student, MIREA - Russian Technological University (RTU MIREA)

Аннотация: В наше время остро стоит вопрос об разработке грамотной архитектуры приложений. Развитие грамотных архитектуры приложений обеспечивает простую разработку и легкую эксплуатацию продукта на долгий период времени. В данной статье мы рассмотрим архитектуру приложений как один из важнейших аспектов для достижения его наилучшей работоспособности.

Abstract: Nowadays, the issue of developing a competent application architecture is acute. The development of competent application architecture ensures simple development and easy operation of the product for a long period of time. In this article, we will consider the architecture of applications as one of the most important aspects to achieve its best performance.

Ключевые слова: Архитектура, приложения, хорошая архитектура, виды архитектуры.

Keywords: Architecture, applications, good architecture, types of architecture.

Настоящая статья посвящена теме что такое архитектура, зачем она нужна, её виды, критерии хорошей архитектуры.

В каждой программе важно не только хорошо написанный код, но и грамотная организация всего проекта. Продуманная архитектура внутри программы имеет значение как в маленьких программах, так и в огромных проектах. Это важно, поскольку чаще всего проект обрастает сложностью быстрее чем четкой структурой организации. Поэтому если не озаботиться вопросом архитектуры приложения заранее, программисты могут очень быстро столкнуться с проблемой контроля разрабатываемого программного приложения. Так же правильно составленная архитектура внутри приложения экономит множество ресурсов, например: деньги, время и силы разработчиков. Каким простым для решения не казалась бы поставленная задача, чаще всего лучшим способом начала работы будет составить её архитектурную модель реализации.

Что такое «Архитектура приложений»? При рассмотрении «Архитектуры приложения», можно выделить две основные составляющие, на которые разделяется данное понятие:

– Дизайн
– Низкоуровневые детали

Каждая из составляющих архитектуры приложения имеет свою роль и свое значения на этапе проектирования приложения. Поэтому рассматривать архитектуру приложения как низкоуровневые детали или дизайн по отдельности в корне неверное суждение.

Для понимания данного разделения проще всего взять архитектуру дома. При рассмотрения цельной конструкции дома, за архитектуру будут приняты такие вещи как: уступы, карнизы, крыша, стены, расположения комнат и организация пространства внутри. То есть дизайн такого объекта как дом. Однако, стоит углубится в чертежную модель, как можно будет увидеть организацию системы проводки, в виде выключателей и розеток, организацию системы отопления в виде батарей и бойлеров. Эта часть внутри дома является низкоуровневыми деталями. Как видно из примера все вышеперечисленные составляющие являются частью архитектуры дома, которые относятся и к низкоуровневым деталям, и к дизайну. Эти составляющие в совокупности образуют единую систему архитектуры, а дом не сможет быть полноценным как без деталей на уровне дизайна, так и без низкоуровневых.

В архитектуре приложения ситуация аналогичная архитектуре дома. В каждой компании, занимающийся разработкой, нужно четко понимать, что такое архитектура приложения, и знать важность каждой из её составляющих. Серьезное отношение различных организаций к архитектуре подразумевает то, что они готовы уделить достаточно много времени на разработку архитектуры приложения, поскольку в будущем это поспособствует уменьшению трудозатрат и повышению эффективности в ходе разработки проекта.

Цели и задачи архитектуры, общие и частные

Для чего используется архитектура приложений? В чем состоит цель хорошего дизайна ПО? По словам Роберта Мартина:

«Цель архитектуры программного обеспечения — уменьшить человеческие трудозатраты на создание и сопровождение системы.»

Поэтому каждая программа, написанная на основе какой-либо архитектурной модели, имеет одну и ту же меру качества дизайна. Этой мерой является совокупность трудозатрат, необходим для погашения запроса, оставленного клиентом. Если система имеет хороший дизайн и внимание к низкоуровневым деталям, в таком случае трудозатраты на эксплуатацию данного программного обеспечения будут невелики. Однако если трудозатраты увеличиваются с каждой последующей версией – это означает что архитектура приложения построена неэффективно и неверно.

Для того чтобы не допустить неверного построения архитектуры приложения, необходимо знать процесс её построения.

Этот процесс можно разделить на следующие основные этапы:

1. Определение целей архитектуры – наличие четкий целей способствует в правильном отборе проблем для решения, помогает разделить разработку на четкие этапы;
2. Выявление основных сценариев – необходимо выявить основные сценарии, для понимания того, что является основной задачей, а что второстепенной;
3. Создания прототипа приложения – необходимо определить архитектуру развертывания, тип приложения и его архитектурные стили;
4. Выявление потенциальных проблем – необходимая установка основных проблем области на основании требований к качеству;
5. Определение вариантов решения – создание в каждой итерации прототипа, который является доработкой или развитием предыдущего решения.

Такой процесс разработки архитекторы предполагает первоначальную разработку дизайна, удовлетворяющего всем требованиям, прописанным, ограничениям и параметрам качества. Впоследствии, в ходе разработки происходит корректировка сценариев, и пересмотр основных целей архитектуры, общего представления о программе и способов решения проблемы.

Цель архитектуры или её задачи определяются для ограничения архитектуры при постановке проблем, которые она решает.

К основным элементам при определении целей и задач архитектуры относятся следующие:

– Первичное определение задач архитектуры – от задач напрямую зависит время, затрачиваемое на разработку готового решения (архитектуры приложения). Также необходимо определить, насколько поэтапной является разработка архитектуры, достаточно ли сделать рабочий прототип и провести тестирования или планируется разработка на долгий срок;

– Определение клиентов архитектуры – необходимо учесть клиентов разрабатываемой архитектуры, для того чтобы сделать разрабатываемую архитектуру приложения максимально удобной для них;

– Определение ограничений архитектуры – ограничения необходимо определить вначале работы для того, чтобы при дальнейшей разработке не сталкиваться с неожиданностями.

В качестве примера возьмем определение целей и задач при разработке приложение для заказов блюда из ресторана:

– задачей архитектуры будет являться разработка нового мобильного приложения для заказа блюд через Интернет;

– потребителями будут тестеры, разработчики и другие ИТ специалисты. Архитектура будет представлена заказчику на одобрение, для внесения дальнейших правок, если в них будет необходимость;

– основные ограничения – приложение разрабатывается только для IOS, отсутствие рекламы и ограничение на разработку и содержание серверной части приложения, устанавливаемые заказчиком.

Виды архитектур приложений и три основных парадигмы

При выборе архитектуры приложения, разработчик опирается на те цели и задачи, которые он должен добиться и которые он должен решить. В

соответствии с многообразием тех целей и задач, с которыми может столкнуться разработчик были созданы множество видов архитектуры. Каждый из видов архитектуры способен решить свой порог целей и задач, и упростить разработчику выявление принципов разрабатываемой архитектуры с нуля.

В настоящие время выделяют следующие 5 основных видов архитектур приложений:

1. Многослойная архитектура – это архитектура, работающая по принципу разделения ответственности за разные компоненты приложения между слоями, которые «лежат» друг на друге и выполняют разные обязанности;

Данный вид архитектуры разделяет ПО на следующие три слоя:

– Presentation layer – отвечает за интерфейс пользователя и обеспечивает пользователю структурированный опыт использования приложения;

– Business logic layer – отвечают за логику приложения, он отделяет дизайн взаимодействия пользователя с приложением, от вычислений, которые в нем проводятся;

– Data link layer – отвечает за взаимодействие приложения с базами данных и другими видами хранилищ большого объема данных.

Данные слои в приложении проходят через каждый его элемент, тем самым влияя на его стабильность и качество разработанного ПО.

2. Многоуровневая архитектура – данный вид архитектуры приложения разделяет ПО на уровни по типу взаимодействия, самая доступная аналогия «Клиент-сервер». В основном архитектуру можно разделить на три подтипа: одноуровневая, двухуровневая, и многоуровневая.

Этот вид архитектуры для взаимодействия между слоями использует принцип «Запрос-ответ» и как следствие имеет большую возможность масштабируемости, в отличии от многослойной архитектуры.

3. Сервис-ориентированная архитектура – данная архитектурная модель состоит из 5 частей, которые связанны между собой.

Таким образом она состоит из следующих 5-ти элементов:

– Services (сервисы)
– Service Bus (сервисная шина)
– Service Repository (сервисный репозиторий)
– SAO Security (САО безопасность)
– SAO Governance (управление САО)

Каждый из этих элементом задействуется в протоколе отправки клиентом запроса и получения им ответа от приложения. Клиент посылает запрос, который в свою очередь обрабатывает SB (основной компонент сервис-ориентированной архитектуры, отвечает за маршрутизацию и оркестровку). С помощью SR (Service Repository) SB создает запрос в специальный сервис, взаимодействующий с другими сервисами или базами данных (управляющий сервис), для того чтобы в последствии вернуть данные (ответ) пользователю.

В общем случае данные сервисы делятся на два вида:

– Atomic services (Атомарные сервисы)
– Compositive services (Композиционные сервисы)

4. Микроархитектура – данный вид архитектуры является противоположностью монолитной. В нем приложение разрабатывается как отдельный набор микросервисов, каждый из которых автономен в своей работе, а между собой они связываются механизмами для совместной работы.

Эти сервисы могут быть развернуты вне зависимости друг от друга, от сюда и высокая модульность проектов на микро сервисной архитектуре. В связи с этим отсутствует централизованный орган управления между сервисами, вернее он присутствует, но его наличие в связи микросервисов между собой минимально.

5. Монолитная архитектура – монолитная архитектура является полной противоположностью микросервисной.

Различные компоненты программы при её разработке объединяются в единую программу на одной платформе. Все части программного приложения централизованы в одном месте управления.

Монолитная архитектура удобна для работы небольших команд программистов или же для работы в команде над небольшим проектом. Эта архитектура является одной из самых первых архитектур для написания приложений, поэтому в настоящее время практически никто не пользуется, предпочитая более удобные микро сервисные или же многослойные архитектуры.

Критерии хорошей архитектуры

Познакомившись с основными видами архитектур, необходимо определится с критериями, по которым будет возможность определить является архитектура «хорошей» или «плохой». Сразу стоит оговорить, что такого понятия как «хорошая архитектура» или же «плохая архитектура» в полноценном понимании не существуют и существовать не могут. У каждой архитектуры есть свои достоинства и недостатки, которые стоит рассматривать при её выборе под решение определенных задач, которые были поставлены перед разработчиком. В реалиях частного случая каждого заказа можно выделить подходящую архитектурную модель или модель, реализация которой попросту не имеет смысла в реалиях решения текущих задач.

Исходя из выше сказано можно выделить следующие основные критерии архитектуры:

– Эффективность системы – показывает, насколько хорошо программа справляется с поставленными для неё задачами и как хорошо она выполняет свои функции, как в условиях, поставленных в ТЗ, так и вне их (надежность, безопасность, производительность, способность справляться с увеличением нагрузки и т.д.).

– Гибкость системы – описывает насколько хорошо приложение способно подвергаться дополнительным изменениям к основному функционалу.

Чем меньше проблем и ошибок вызовет внедрение новых функций, тем более гибкой можно считать систему и наоборот. Для соблюдения данного критерия внутри архитектуры необходимо учитывать, что каждый компонент системы не должен влиять на другие.

– Расширяемость системы – критерий, оценивающий насколько удобно добавлять в систему новые функции, не нарушая её целостности и основной части. На начальном этапе разработки, необходимо реализовывать только начальный функционал, но при этом архитектура с данным критерием должна иметь возможность его легкого расширения.

Данные критерии объединены в отдельный блок принципов: «Принцип открытости/закрытости» («Open-Closed Principle» - один из 5-ти принципов SOLID), все элементы программы должны бить открытыми для расширения, но закрытыми для модификации.

– Масштабируемость процесса разработки – характеристика, которая отражает возможность сокращения сроков работы над проектом из-за внедрения в неё новых специалистов.

– Тестируемость – показывает, насколько легко тестировать проект команде тестировщиков и как следствие отражает то количество ошибок и недочетов в коде, которые влияют на работу системы в целостности. Требование хорошей тестируемости в программе является критерием, который приводит программу к хорошему дизайну и позволяет оценить его качество.

– Возможность повторного использования – показатель возможности использования фрагментов приложения в ходе разработки других приложений.

– Структурированный код и сопровождаемость – эти критерии можно отнести к одной группе так как один из них напрямую влияет на другой. Поскольку разработка программы — это длительный процесс в ходе него происходят кадровые изменения ряда специалистов, и для того, чтобы каждый из смененных специалистов мог оперативно присоединиться к разработке ПО

или же к его поддержке, необходимо чтобы все элементы имели хорошо структурированный код, без дублирования и разумеется документацию.

Список используемых источников

1. Роберт Мартин Чистая архитектура. Искусство разработки программного обеспечения : учебное пособие / Роберт Мартин; Санк-Петербург : Издательский дом «Питер», 2018. - 352 с. - ISBN 9785446107728, 5446107721. - URL: https://vk.com/wall-51126445_24407 (дата обращения: 20.04.2022). - Режим доступа: vk.com. Текст: электронный
2. Создание архитектуры программы или проектирование табуретки - URL: https://habr.com/ru/post/276593/ (дата обращения: 25.04.2022). - Режим доступа: Агрегатор статей habr. Текст: электронный.
3. Монолитная архитектура. Традиционный метод разработки приложений - URL: https://codernet.ru/articles/drugoe/monolitnaya_arxitektura_tradiczionnyij_metod_razrabotki_prilozhenij/ (дата обращения: 19.04.2022). – Режим доступа: Агрегатор статей CoderNet. Текст: электронный
4. Архитектура программного обеспечения - URL: https://studref.com/320287/informatika/arhitektura_programmnogo_obespecheniya (дата обращения: 18.04.2022). - Режим доступа: Агрегатор статей StudRef. Текст: электронный.
5. Архитектура программы - URL: https://steptosleep.ru/архитектура-программы/ (дата обращения: 21.04.2022). – Режим доступа: Агрегатор статей StepToSleep. Текст: электронный.

List of sources used

1. Robert Martin Pure Architecture. The art of software development : a textbook / Robert Martin; St. Petersburg : Publishing House "Peter", 2018. - 352 p. - ISBN 9785446107728, 5446107721. - URL: https://vk.com/wall-51126445_24407 (accessed: 04/20/2022). - Access mode: vk.com . Text: electronic

2. Creating a program architecture or designing a stool - URL: https://habr.com/ru/post/276593 / (accessed: 04/25/2022). - Access mode: Article Aggregator habr. Text: electronic.
3. Monolithic architecture. The traditional method of application development is URL: https://codernet.ru/articles/drugoe/monolitnaya_arxitektura_tradiczionnyij_metod_razrabotki_prilozhenij / (accessed: 04/19/2022). – Access mode: Article aggregator CoderNet. Text: electronic
4. Software Architecture - URL: https://studref.com/320287/informatika/arhitektura_programmnogo_obespecheniya (accessed: 04/18/2022). - Access mode: StudRef Article Aggregator. Text: electronic.
5. Program architecture - URL: https://steptosleep.ru/архитектура-программы / (accessed: 04/21/2022). – Access mode: StepToSleep article Aggregator. Text: electronic.

Для цитирования: Бежик А.А., Мажей Я.В. АРХИТЕКТУРА ПРИЛОЖЕНИЙ. ЧЕМ ЯВЛЯЕТСЯ? ДЛЯ ЧЕГО ИСПОЛЬЗУЕТСЯ? ОСНОВНЫЕ ВИДЫ И КРИТЕРИИ ХОРОШЕЙ АРХИТЕКТУРЫ// Научный сетевой журнал «Столыпинский вестник» №9/2022