Модерно Банково Управление

(ABM)

Документ за софтуерната архитектура

Версия 1.5

История на промените

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 23/01/2016 | 1.0 | Създаване на документа | Адриан Данаилов, Мартин Абрашев, Калоян Гецов |
| 24/01/2016 | 1.1 | Разработване на чернова | Калоян Гецов |
| 24/01/2016 | 1.2 | Добавяне на информация | Мартин Абрашев |
| 25/01/2016 | 1.3 | Добавяне на информация | Адриан Данаилов |
| 26/01/2016 | 1.4 | Добавяне на информация | Калоян Гецов, Адриан Данаилов |
| 28/01/2016 | 1.5 | Завършване на архитектурата | Мартин Абрашев |

Съдържание

[1. Въведение 5](#_Toc441778633)

[1.1 Цел 5](#_Toc441778634)

[1.2 Обхват 5](#_Toc441778635)

[1.3 Дефиниции, Акроними и Абревиатури. 5](#_Toc441778636)

[1.4 Препратки 5](#_Toc441778637)

[1.5 Кратък преглед 5](#_Toc441778638)

[2. Представяне на архитектурата 5](#_Toc441778639)

[2.1 Архитектурни цели и ограничения 5](#_Toc441778640)

[2.2 Функционален поглед 5](#_Toc441778641)

[2.3 Логически поглед 5](#_Toc441778642)

[2.4 Интеграционен поглед 5](#_Toc441778643)

[2.5 Поглед над хардуерните компоненти 5](#_Toc441778644)

[2.6 Имплементационен поглед 6](#_Toc441778645)

[2.7 Качество 6](#_Toc441778646)

[3. Архитектурни цели и ограничения 6](#_Toc441778647)

[3.1 Бюджет 6](#_Toc441778648)

[3.2 Наличие на “legacy” код 6](#_Toc441778649)

[3.3 Географско разположение на системата 6](#_Toc441778650)

[3.4 Обем на данните 7](#_Toc441778651)

[3.5 Връзка с чужди системи 7](#_Toc441778652)

[3.6 Сигурност 7](#_Toc441778653)

[3.7 Устойчивост на данните 7](#_Toc441778654)

[3.8 Използвани технологии 7](#_Toc441778655)

[3.8.1 GIT 7](#_Toc441778656)

[3.8.2 Maven 8](#_Toc441778657)

[3.8.3 Hibernate 8](#_Toc441778658)

[3.8.4 Apache Tomcat 8](#_Toc441778659)

[3.8.5 Web услуги 8](#_Toc441778660)

[4. Функционален поглед 9](#_Toc441778661)

[4.1 9](#_Toc441778662)

[5. Logical View 9](#_Toc441778663)

[5.1 Overview 9](#_Toc441778664)

[5.2 Architecturally Significant Design Packages 9](#_Toc441778665)

[5.3 Use-Case Realizations 9](#_Toc441778666)

[6. Process View 9](#_Toc441778667)

[7. Поглед над хардуерната инфраструктура 9](#_Toc441778668)

[8. Имплементационен поглед 10](#_Toc441778669)

[8.1 Overview 10](#_Toc441778670)

[8.2 Layers 10](#_Toc441778671)

[9. Качество 10](#_Toc441778672)

Софтуерна архитектура

# Въведение

## Цел

Целта на този документ е да опише софтуерната архитектура на ABM. Той покрива както хардуерa, който е нужен за пълното и функциониране, така и софтуера който ще се използва за реализацията на различните под системи. Архитектурата дава ясна представа на екипа разработващ системата какви стъпки трябва да поеме той, както и предотвратява възможността от възникването на по – нататъшни проблеми за системата.

## Обхват

Документът обхваща хардуерът, софтуерът както и част от модулната реализация на системата.

## Дефиниции, Акроними и Абревиатури.

* Информация за тях може да бъде намерена в документа „ABM-4-I1-Glossary“.

## Препратки

В този документ ще има препратки към следните документи:

* ABM-I1-5-SoftwareRequirementSpecifications.doc
* ABM-E1-2-Infrastructure Model.docx

## Кратък преглед

В следващите точки се разглеждат основните фактори, които се взимат в предвид при изграждането на една система и решенията на проблемите които повдигат те. Технологиите планирани за използване при реализацията и.

Във втората част от документа ще се разгледат различните погледи над системата съпътствани с диаграми и кратки обяснения.

# Представяне на архитектурата

Софтуерната архитектура е разработена, така че да съдържа различни погледи над системата.. Следва списък с описание на различните гледни точки, чрез които е изграден документа.

В отделните глави има подробно описание на различните погледи над системата, включващо и на взетите решения, обосновката зад тях и разяснителни диаграми.

## Архитектурни цели и ограничения

Това е секцията, в която се описват целите и ограниченията стоящи пред архитектурата. За да бъде надеждна тя трябва да удовлетвори целите, изискванията към системата и да се съобразява с поставените пред нея ограничения. В нея също са обяснени и използваните технологии.

## Функционален поглед

Преглед на основните подмодули и информационни обекти, които ще се използват в системата.

## Логически поглед

Поглед над избраната архитектура, компонентите на системата и комуникацията между тях. Разискват се избраните архитектурни шаблони, разбивката на подмодули, връзката и целите на всеки от тях.

## Интеграционен поглед

Връзката с външни системи е важна част от системата и поради това в тази точка се описва логиката, на която са реализирани.

## Поглед над хардуерните компоненти

Хардуерен поглед над системата.

## Имплементационен поглед

Компонентите и разположението им по слоеве. Как отделните компоненти са вързани с другите, как комуникират помежду си.

## Качество

С какво е допринесла системата в облекчаването на работата спрямо сегашната. Бъдещо развитие.

# Архитектурни цели и ограничения

Архитектурата се построява като се имат впредвид функционалните и нефункционалните изисквания описани в спецификацията на допълнителните изисквания.

Включват се ограниченията, наложени от средата, в която ще работи софтуерната система и налагане на определни стандарти (например за интеграция с външни системи).

Поради характера на проекта се налага използването на безплатен и/или софтуер с отворен код. Не е предвидено закупуването на готови системи, които да бъдат интегрирани и да ускорят процеса на работа. Този начин на работа налага използването на програмния език Java и платформата за разработка Eclipse. Проекта представлява разработването на уеб приложение и ще се използват съответно EE (Enterprise Edition) версиите на Java и Eclipse.

Отделните модули са оформени като отделни java проекти. За нуждите на проекта е създадено GIT хранилище, което съдържа актуална версия на проектите.

За изготвянето на архитектурата на тези система са взети под предвид следните фактори:

* Софтуерните изисквания, описани в документа за спецификация на допълнителните изисквания, зададени от възложителя на проекта;
* Нефункционални изисквания, описани в гореспоменатия документ;
* Географското разположение на системата;
* Обемът на данните с който системата работи;
* Нуждата от връзки с чужди системи;
* Бюджета нужен за реализация на проекта;
* Живот на системата;
* Наличие на “Legacy” код;
* Подобни системи.

## Бюджет

Взимайки в предвид възможностите на фирмата разработчик и договора сключен с фирмата възложител технологиите и хардуерът нужен за пълното осъществяване на проекта е комбинация от платени и безплатни услуги. Поради силната зависимост от хардуерни технологии на системата тук място за компромис не се допуска. В точки 2.2 до точка 4 са описани всички технологии нужни за реализация на системата.

## Наличие на “legacy” код

Създаването на тази система ще се осъществи без наличен код на”legacy” код от предходни системи, който програмистите да надграждат. Това означава, че фирмата изпълнител поема пълна отговорност за жизнения цикъл на системата и поведението и. Въпреки липсата на “legacy” код голяма част от функционалностите на системата стъпват на чужди разработки и технологии. Банкоматните системи са добър пример за чужда разработка, тук единствено трябва да се контролира диалога между двете системи за да се гарантира тяхната ефективност.

## Географско разположение на системата

Взимайки в предвид големината на проекта и разположението на системата в цяла Европа. Топологията на мрежата, която отговаря най – добре на нашите цели е “Mesh network”. Тя предлага най – надеждна връзка до всяка една точка като потенциално може да загуби скорост за сметка на успешно доставени данни.

Осъществяването на вътрешните мрежи ще се осъществява с топология от тип шина, която предлага бързина, надеждност и ниски разходи. Централният клон ще играе ролята на главен управленчески орган, при настъпване на промени по системата той ще изпраща информация към по – малките клонове.

## Обем на данните

Според количеството на данните който ще постъпват и ще се обработват се налага те да се съхраняват с “Data center” технологии. Тази технология позволява съхранението на данните на един сървър, който предлага високо ниво сигурност спрямо пробив на данните, както и създаване на архив на файловете. Една от най – важните му характеристики е способността да се надгражда което позволява обемът на съхранените да надвишава далеч над нормалното.

Поради спецификата на данните и разграничаването на различните нива на достъп системата ще ползва централизиран модел на управление. Това включва забавяне при изпълнение и обновяване но добавяне на допълнително ниво сигурност и контрол над системата.

## Връзка с чужди системи

Поради спецификата на работата на тази система се налага осъществяването на връзки с други системи. Най – наложилият се принцип на осъществяване на връзки с други системи е чрез сервизна шина(Enterprise Service Buss). Тя не само позволява общуването между различни приложения и системи, но и надеждно ниво на защита. Чрез нея ще може да се осъществят връзки като с държавни системи така и с други банки или “smart phone” приложения. Въпреки завишените разходи за поддръжката и на фона на конкурентността на други фирми използването и е задължително.

## Сигурност

Основен компонент на системата е сигурността. За осигуряването на контрола на достъп ще бъде разработен модул, които да позволява подържането на информация за регистрираните (оторизирани) потребители. Модула ще е базиран на Spring Security framework. Достъпа до системата може да се осъществи само от оторизирани потребители, а конкретните права за достъп до ресурсите се контролират от ролите на потребителя.

Следните възможности на системата трябва да се разработят:

* Аутентификация – влизане в системата с потребителско име и парола
* Оторизация – достъп до ресурсите за различните потребители, задава се посредством роли.
* Конфиденциалност – възможност за съхраняване и пренасяне на данните в криптиран вид.

Компютрите в различните клонове на банката ще се достъпват чрез VPN. Това ще добави допълнително ниво на сигурност към системата, като позволява ползването на ресурси по мрежата единствено чрез попълване на акаунт и парола и преминаване на проверка за MAC адрес.

Виртуализацията намира своето приложение и извън сигурността на системата. Взимайки в предвид мащаба на системата се очаква голям поток от потребители, както и пикове и спадове на натовареност на системата. Чрез виртуализация ще се ограничи използването на машини, като те ще влизат в употреба само тогава, когато са нужни.

## Устойчивост на данните

В проекта ABM информацията ще се съхранява под формата на база от данни и файлова система.

Основен приоритет е информацията да се запазва по надежден начин, гарантиращ целостта и устойчивостта й.

Съхранените данни трябва да могат да бъдат достъпвани, модифицирани и добавяни по определени от проекта критерии. Използвания DBMS ще е MySQL, за драйвер ще се ползва MySQL Connector/J driver.

## Използвани технологии

### GIT

GIT е безплатна система, с отворен код, за контрол на версийте. GIT е предназначен да се справя бързо и ефективно, както с малки проекти, така и с проекти с огромни размери. Системата е лесна за научаване, изисква малко ресурси и е светкавично бърза. Надминава съперниците си в този клас – Subversion, CVS, Perforce и други, с функционалности като евтина поддръжка на клони (cheap local branching), удобна зона за зареждане на файлове преди запис (convenient staging areas) и други.

### Maven

Maven е инструмент за автоматизиране на билдовете при работа с Java проекти. Работи чрез използването на XML файл, в който е описана информацията вънпните зависимости на билда (external dependencies). Инстурментът позволява контрол над реда на билдване, директорийте и плъгините.   
Друга част от функционалностите на Maven е възможността да се свалят динамично външни API, библиотеки, модули, и други, като след сваляне те се съхраняват в локално хранилище.

В текушия проект, в средата за разработка, на сървъра е създадено хранилище, което да се използва за стабилните версии на подмодулите и за пускане на билдове.

### Hibernate

Hibernate дава на разработчиците възможност по лесно да разработват приложения, чиито данни надживяват процесите на приложението. Като Object/Relational Mapping (ORM) framework, Hibernate се занимава с персистирането на данните, което прави с релационите бази (чрез JDBC).

Като допълнение към собствения си API, Hibernate също е имплементация на Java Persistance API (JPA). Като такова, то може лесно да бъде използвано от всяка среда поддържаща JPA, включително Java SE приложения, Java EE приложни сървъри, и други.

Hibernate помага на разработчиците да разработват персистентни класове, като следва обектно-ориентираните идиоми, включително наследяване, полиморфизъм, асоциация, композиция и други. Hibernate не изисква интерфейси или базови класове за персистентните класове и структури, и поддържа възможността всеки клас или структура да стане персистентен.  
Системата използва ORM технологията Hibernate за работа с данните.

### Apache Tomcat

Apache Tomcat (наричан често Tomcat) е web сървър с отворен код, разработен от Apache Software Foundation. Той имплементира доста от Java EE спецификациите, включително Java Servlet, Java Server Pages (JSP), Java EL и WebSocket и предоставя „чисто Java” HTTP web сървър, на който да се публикуват Java приложения.

Сървърите, на които са публикувани сървисите и Java Web системата са приложни сървъри, на които е инсталиран Apache Tomcat.

### Web услуги

В рамките на проекта ще се използват уеб услуги, за комуникация между системите с бизнес логика и слоя за данни, както и за интеграция с външни системи. Web услугите представляват функционалности, достъпни на даден адрес в мрежата или в облак. Те предоставят стандартни интерфейси за комуникация между системите. Тези стандарти се основават на технологии, които са общи за уеб приложенията като: XML, JSON, HTTP.

В рамките на проекта се използва framework-a на Java Spring, който предлага библиотеки и шаблони за създаването на REST услуги.

# Функционален поглед

Фукционалностите на системата са разделени на модули, като всеки модул представлява съвкупност от логически групирани функционалности. В разработването им е важно да се постигне високо ниво на независимост (да са слабо свързани), с цел лесното заменяне функционалности и поправяне на дефекти (промяна на едно място няма да се отрази на другите). Модулите на системата включват:

* Модул „Управление на влогове, депозити и кредити“;
* Модул „Банкомати“;
* Модул „Управление на карти“;
* Модул „Електронно банкиране“;
* Модул „SMS банкиране“;
* Модул „Управление на инвестиции“;
* Модул „Работа с клиента“;
* Модул „Интеграция“;
* Модул „Сигурност“;

## 

# Логически поглед: архитектура

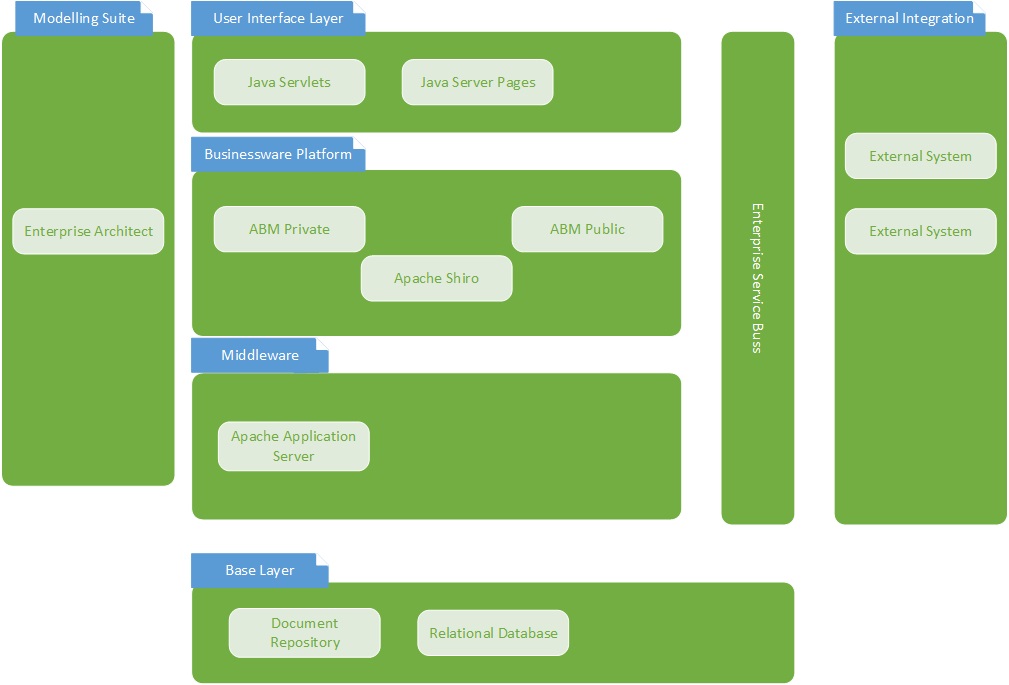
## Цялостна архитектура

Системата ABM представлява разширено трислойно приложение, чиито логически слой се свързва чрез ESB със слоя за данни и външни “service” приложения. Използването на този тип приложения ще подобри системата в следните отношения:

* **Платформена независимост** – Предложените услуги може да се ползват, като на нас ни трябва единствено типът данни, който те очакват да получат и типа данни, който те връщат. Средата на разработка на услугите не ни интересува.
* **Индиректен достъп до услугите** – Понеже услугите се достъпват през ESB тяхното местоположение на сървъра е без значение, понеже шината гарантира тяхната използваемост без значение къде се намират те.

## ABM платформа

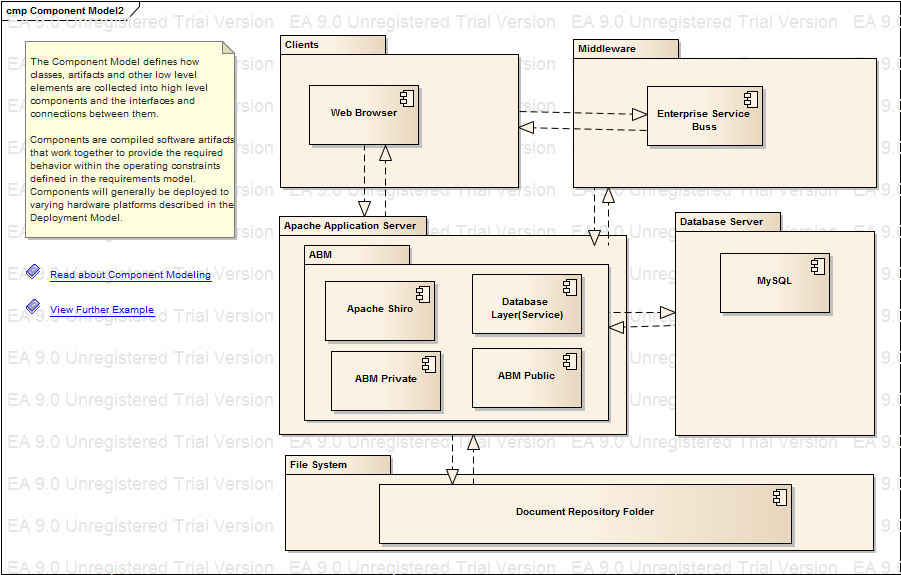
На следващата диаграма е представен общия изглед на платформата. Тук се вижда ясно взаимоотношенията между различните слоеве в системата.

 Описание на отделните елементи:

|  |  |  |
| --- | --- | --- |
| Java Faces Server | Java технология за компонентно създаване на потребителски интерфейси за уеб приложения. |  |
| Java Servlets | Java технология, позволяваща обработката на HTTP заявки и връщането на отговор. |  |
| Java Server Pages | Java технология за динамично генериране на уеб страници. |  |
| ABM private | Ядрото на системата ABM. То се ползва от банковите служители, предлага уникални услуги , включващи модификации на голям брой полета от бази данни, преглед на потока данни с чужди системи и административен контрол над системата. |  |
| ABM public | Втората половина от ядрото на ABM. То наследява частични функционалности на служителските права. Като единствено клиентите работят с него. Предоставя възможността за изпълняване на заявки към БД и създаване на запис в някой полета, съпроводени от административна дейност. Презентационния слой е силно ограничен при този вид потребител. |  |
| Apache Shiro | Управление на данните за потребители, аутентикацията и оторизацията в системата. Може да се интегрира с база от данни или LDAP. | <http://shiro.apache.org/> |
| Apache Application Server | Приложен сървър, в който се стартира приложението, съдържа голям брой готови компоненти. | <http://www.jboss.org/jbossas> |
| Document Repository | Хранилище за документите, което представлява дърво от файловата структура. |  |
| Relational Database | Релационна база от данни. |  |
| Enterprise Architect | Инструмент за UML моделиране. | <http://www.sparxsystems.com/products/ea/> |

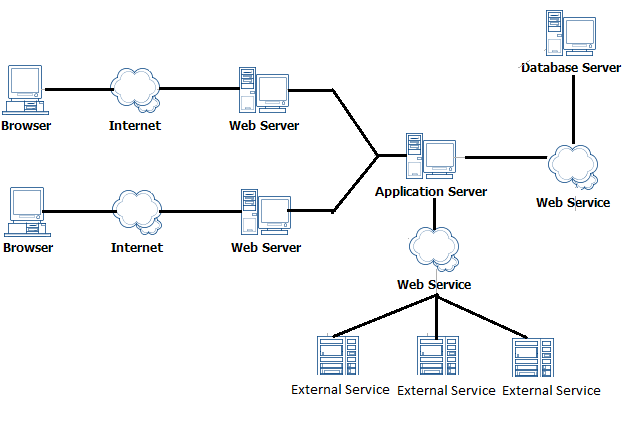
## Сервизни компоненти

В тази точка ще се представят отделните компоненти на системата, като се има в предвид, че някой от технологиите използвани са вече създадени от други фирми с цел безплатно ползване и интеграция с всички платформи.



# Интеграционен поглед

Общ поглед на системата.



В следната картина е изобразена цялостната архитектура на системата. Може да се видят два уеб портала, един за клиенти и един за банкови служители. Те се различават в правата си на достъп до системата. След това се вижда сървърът за приложения, който ще се спомене в следващата точка. В него е разположена цялата логика от приложния слой. Към него са направени връзки чрез ESB шина към “data center” и външни системи.

Поради характера на проекта е много важно системата да бъде разгледана на ниво интеграции с външни системи. ABM ще осъществява връзка с различни системи. Всяка външна система ще предава данни, които може и да не са в един и същи формат. Най-общо интеграциите с външни системи ще изглеждат по следния начин:

**sd Integrations View**

**Външна система 1**

**АВМ**

**Външна система 2**

**Сервизна шина**

**Външна система n**

**Клиент/Потребител**

«flow»

«flow»

«flow»

«flow»

«flow»

«flow»

«flow»

«flow»

«flow»

«flow»

## Общ поглед

АВМ ще осъществява връзка с n на брой външни системи. Връзката в повечето случаи се иницира от АВМ системата към външните приложения. Rest Sеrvice отговаря за интеграцията с външните системи.

## Заложени стандарти

Комуникацията между отделните части ще става с помощта на утвърдени стандарти в уеб услугите. Данните ще се подават под формата на xml със строго определен формат и ще се криптират. Комуникацията става през http/https.

## Сигурност

За всяка система, искаща интеграция с централната деловодна система ще е нужно при изпращане на заявката да се добави Spring Security. Това е разширение, което позволява комуникацията да използва различни средства за сигурност като:

* SAML
* Kerberos
* X.509

Основният фокус е да се използва XML подписване и криптиране, които да осигурят високо ниво на сигурност от точка до точка. Отчасти избрания подход за реализиране на уеб услуги е подбран заради надежността и сигурността, които тези методики позволяват. На всяка въшна система ще бъде предоставен token, чрез който ще се осъществява сигурност на ниво услуга.

# Поглед над хардуерната инфраструктура

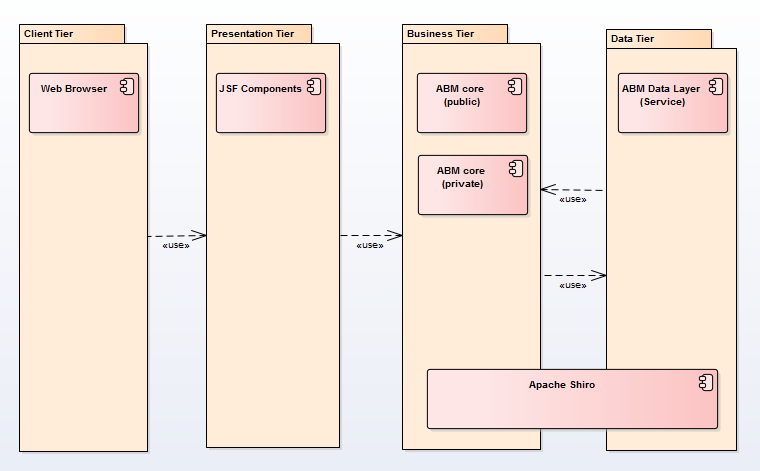
Системата ще работи с шест сървъра, които ще позволят пълното и функциониране. Сървърите включват:

* Вътрешен уеб сървър – VPN осъществяващ контакт между различните машини в мрежата на банката, включва клонове и централен офис;
* Външен уеб сървър – Портал прихващащ идващите от клиентите заявки и пренасочването им към съответната дестинация;
* Два сървъра за приложения – Позволяват лесно общуване между различни компоненти на системата и допринасят към сигурността и.
* Сървър за БД – Това е “Data center” за нашата система, всяка една заявка минава през него. Съдържа информация за всеки един клиент, договор, предлагани оферти и тн.
* Резервен БД сървър – Това е резервния “Data center”, при авария или неизправност с основния сървър, резервния продължава неговата работа до отстраняване на повредата. Той се обновява постоянно спрямо главния сървър, за да може да продължи неговата работа в реално време.
* Сървър архив – При приключване на жизнения цикъл на дадено парче информация то се архивира в отделен сървър и цел премахване на ненужната информация от системата. Сървърът също се намира в “Data center” но на различни машини от тези с актуалните данни.

По - задълбочен поглед над инфраструктурата на системата може да се види в инфраструктурния файл: ABM-E1-2-Infrastructure Model.docx

# Имплементационен поглед

В тази точка се разглеждат разположението на различните компоненти на системата по различните слоеве. Комуникацията започва от потребителя, който през браузъра в клиентския слой комуникира с потребителския интерфейс от презентационния слой. В бизнес слоя се намират функционалностите на системата, свързани с обработката на данни и заявки, както и модулът за оторизация и аутентикация. След като данните бъдат обработени, те се записват в базата, като за това се използва слоя за данни.



# Качество

За системата ABM се идентифицират следните цели за качество:

Стабилност:

* **Описание:** Поведението на системата при увеличаване на потребителските заяви
* **Решение:** Виртуализацията позволява стартирането на допълнителни сървъри без оскъпяване

Надеждност, Наличност:

* **Описание** : Механизъм за преодоляване на срив в системата.
* **Решение :** Сървърите за приложения позволяват бърза смяна с резервните сървъри.

**Преносимост**:

* **Описание** : Способността за повторна употреба
* **Решение :** Системата трябва да е напълно съвместима с приложните сървъри за да може тя да се прехвърли във всеки един момент на друг такъв сървър.

**Сигурност**:

* **Описание**: Механизми за автентикация и оторизация
* **Решение:** Стандартните мерки на сигурност на приложните сървъри ще се приложат и за нашето приложение.