



ТЕХНОСЕРВ
КОНСАЛТИНГ



Обучение сотрудников

Расписание занятий:

День 1-День 3

10:00 – 12:00 Теоретическая часть
12:00 – 16:00 Работа с литературой
16:00 – 18:00 Самостоятельная работа

День 4 – день самостоятельной подготовки

День 5 – реализация итогового практического задания

День 6 – сдача итогового задания (практическая часть)

10:00 – 12:00 Экзамен по практической части

День 7 – сдача итогового задания (практическая часть)

10:00 – 12:00 Экзамен по практической части

День 8 – сдача теоретического экзамена

10:00 – 12:00 Экзамен по теории

План обучения

1-й день:

- Жизненный цикл программного обеспечения
- ★ Виды требований: функциональные и нефункциональные требования. Анализ требований по Вигерсу
- Основные методологии разработки (WaterFall, Agile: SCRUM, Kanban)
- Диаграммы как инструмент анализа требований. Обзор диаграмм UML.
- ★ UseCase диаграмма
- ★ Sequence диаграмма
- ★ Deployment диаграмма
- ★ Activity диаграмма

2-й день:

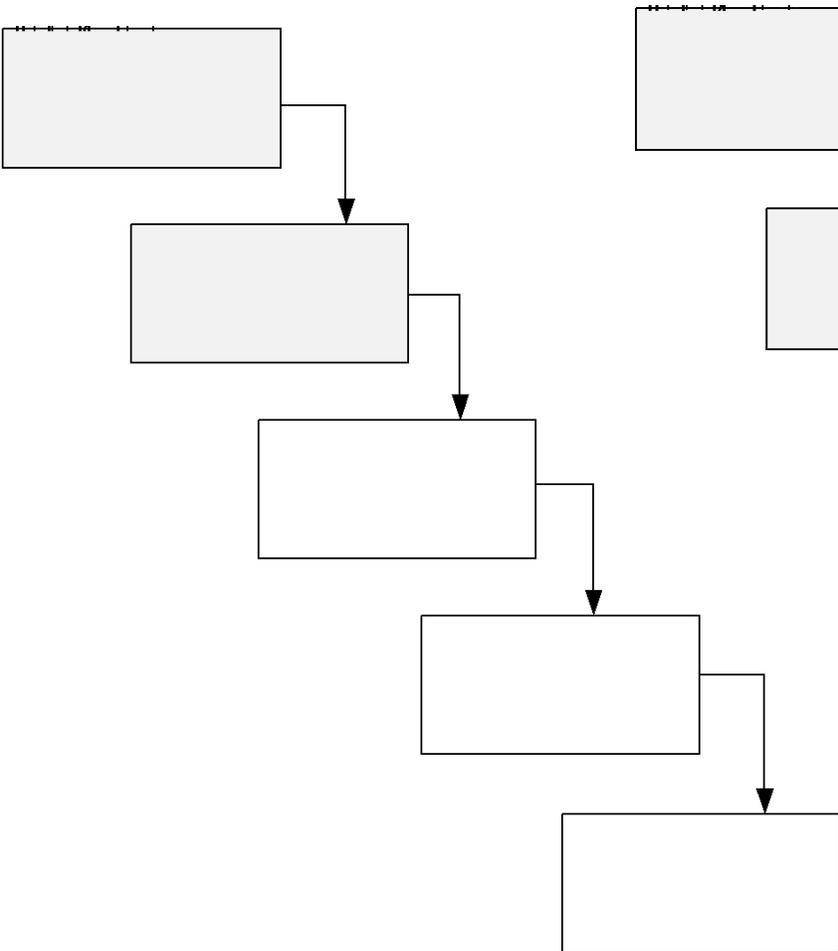
- Протоколы взаимодействия между системами (SOAP, HTTP, Rest Full Api)
- ★ Формат передачи данных между системами (XML:XSD, JSON:Json Scheme, CSV)
- ★ Способы интеграции между системами (ESB, MQ, API w\s, таблицы в БД, файлы)
- Микросервисная архитектура

3-й день:

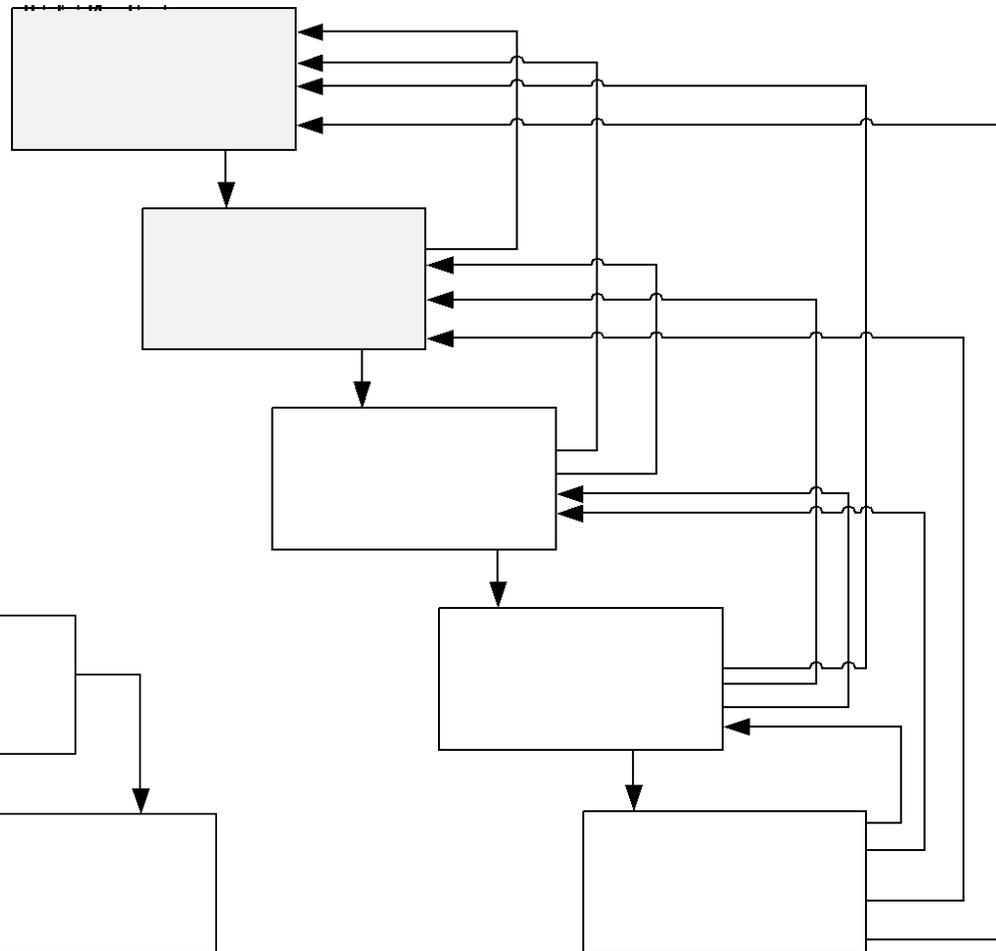
- ★ Структура документа «Техническое задание»
- SMART принципы

Жизненный цикл ПО

Водопадная



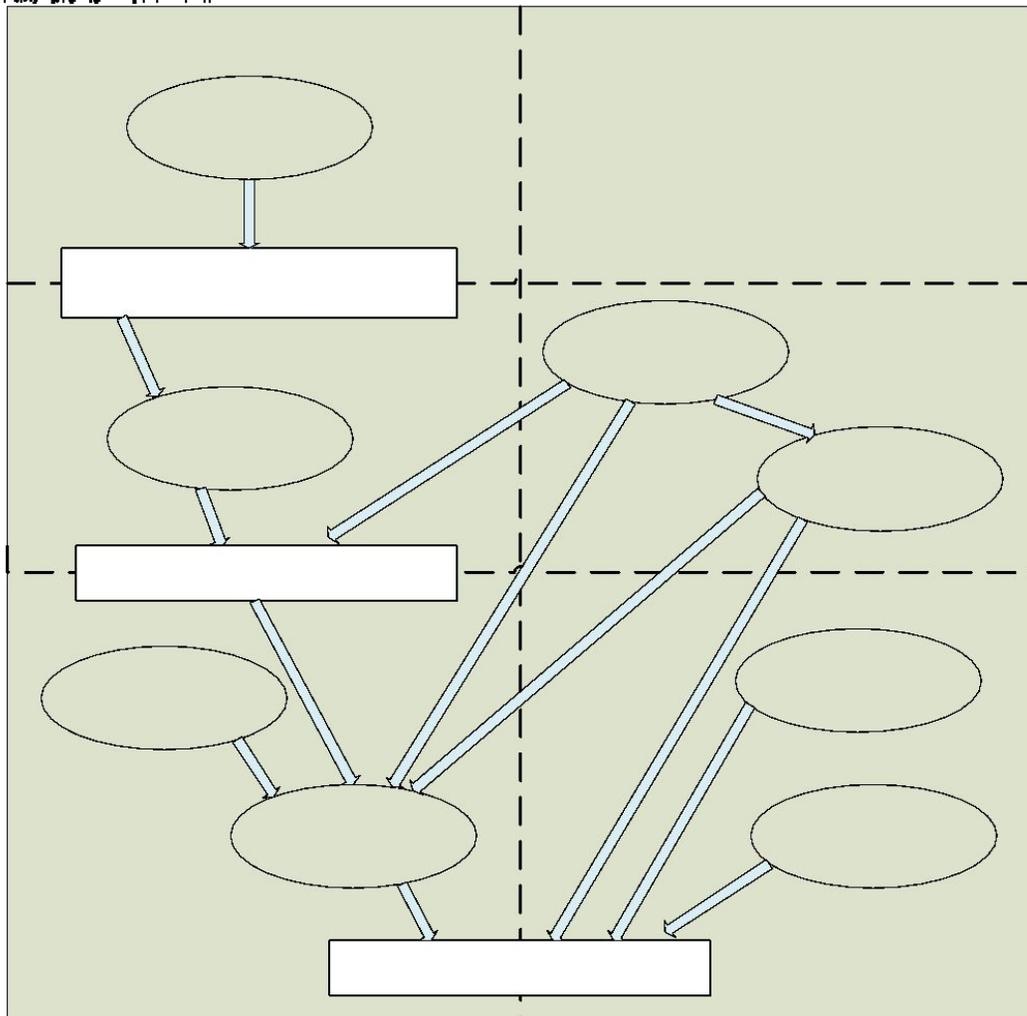
Итерационная



Спиральная



Виды требований



Бизнес требования: описывают высокоуровневые требования к системе. Оформляется в уставе проекта.

Бизнес правила: описывают политики, внутренние регламенты, нормативно правовые акты и т.д.

Требования пользователей: требования о вариантах использования системы

Атрибуты качества: представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков.

Внешние интерфейсы: содержит описание интерфейсов взаимодействия проектируемой системы с внешними системами

Ограничения: описываются рамки используемых инструментов, алгоритмов реализации, структуры продукта

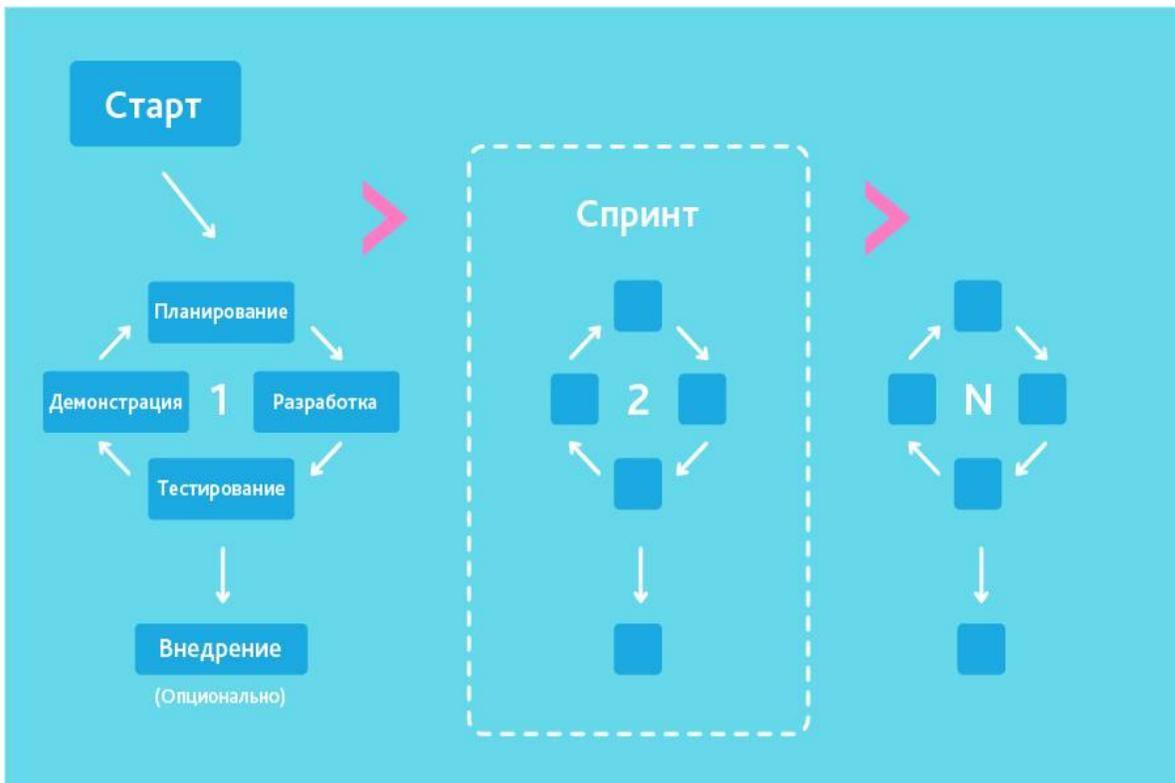
Системные требования: требования к продукту, содержащему несколько подсистем, описывают то, как эти подсистемы должны между собой взаимодействовать.

Функциональные требования: описывают то, как должна функционировать система. Все, что не попало в остальные требования.

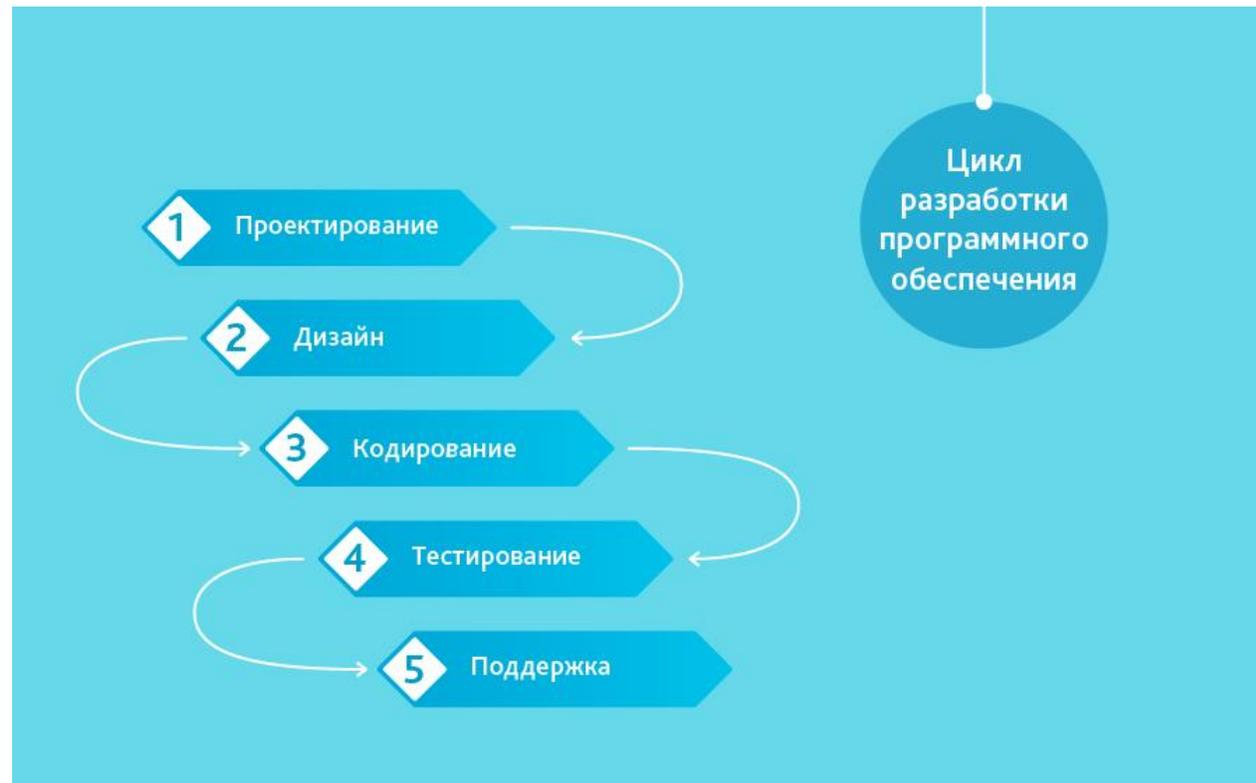


Основные методологии разработки

Agile

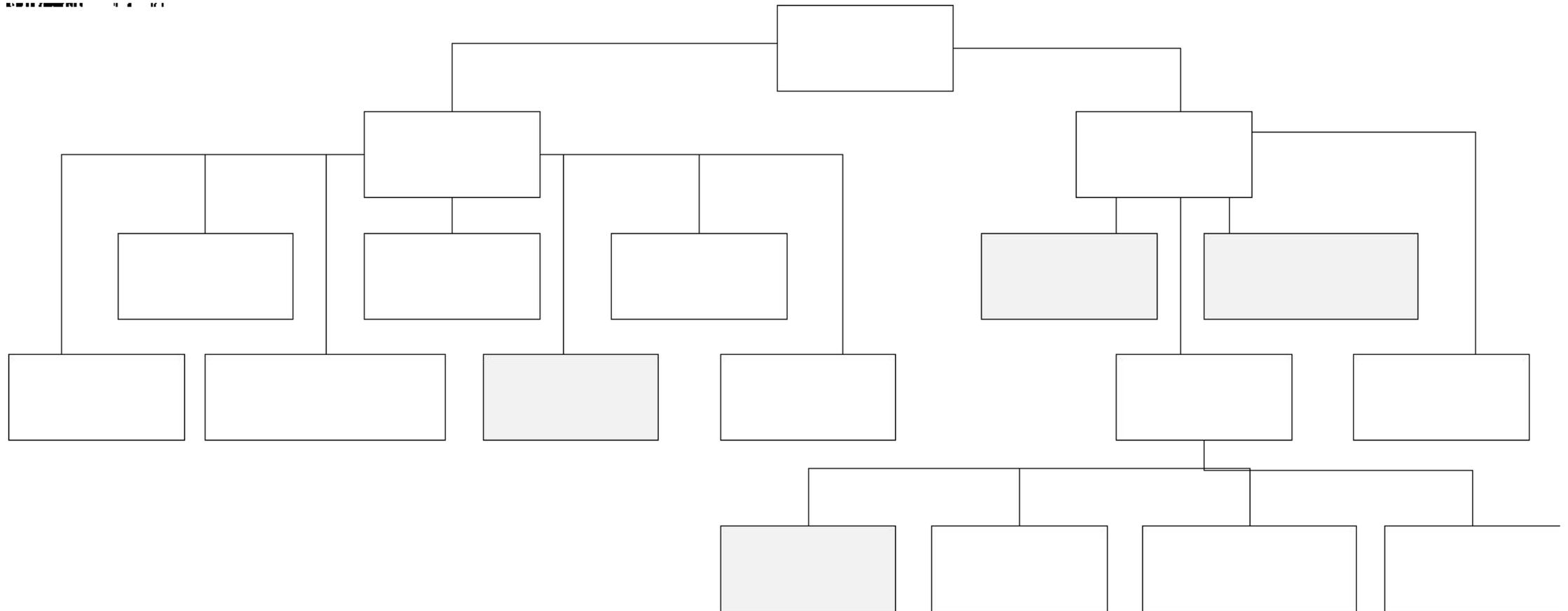


Waterfall



- Люди и взаимодействие важнее процессов и инструментов.
- Работающий продукт важнее исчерпывающей документации.
- Сотрудничество с заказчиком важнее согласования условий контракта.
- Готовность к изменениям важнее следования первоначальному плану.

UML - Unified Modeling Language

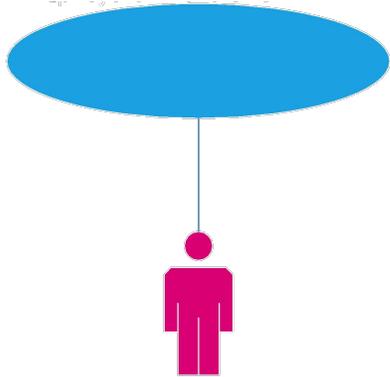


Use Case диаграмма

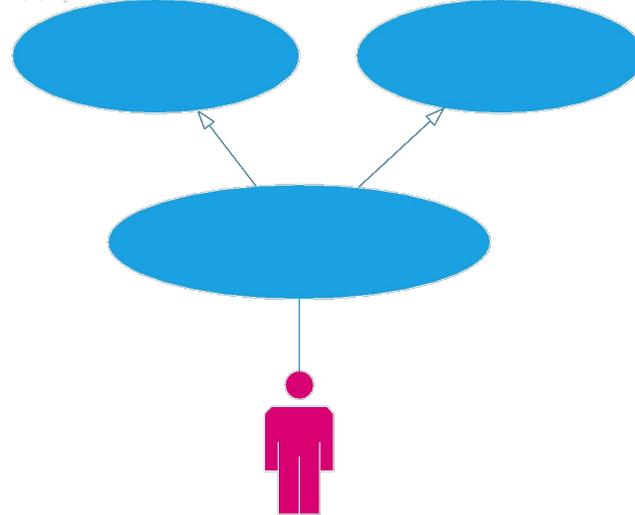
Подсистема



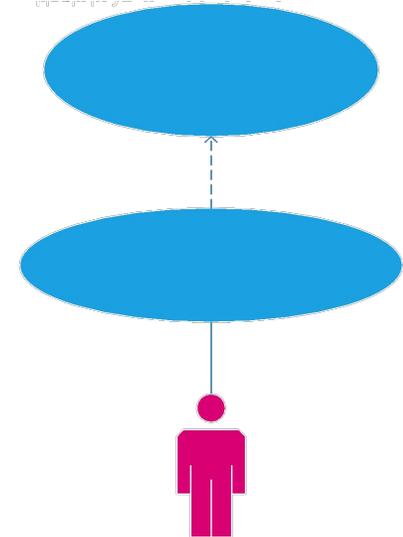
Служит для обозначения
специфической роли актера
в отдельном варианте
использования



Отношение обобщения служит
для указания того факта, что
некоторый вариант использования
А может быть обобщен до
варианта использования В



Указывает, что некоторое
заданное поведение для одного
варианта использования
включается в качестве составного
компонента в последовательность
поведения другого варианта
использования.



Sequence диаграмма



Объект системы, является инициатором взаимодействия

Линия жизни объекта, пока объект существует в системе, указывается его линия жизни



Фокус управления – показывает активное состояние объекта



Применяется как операнд if\then\else



Синхронное сообщение между объектами



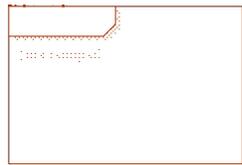
Возвращаемое сообщение. Ответ на запрос



Асинхронное сообщение между объектами



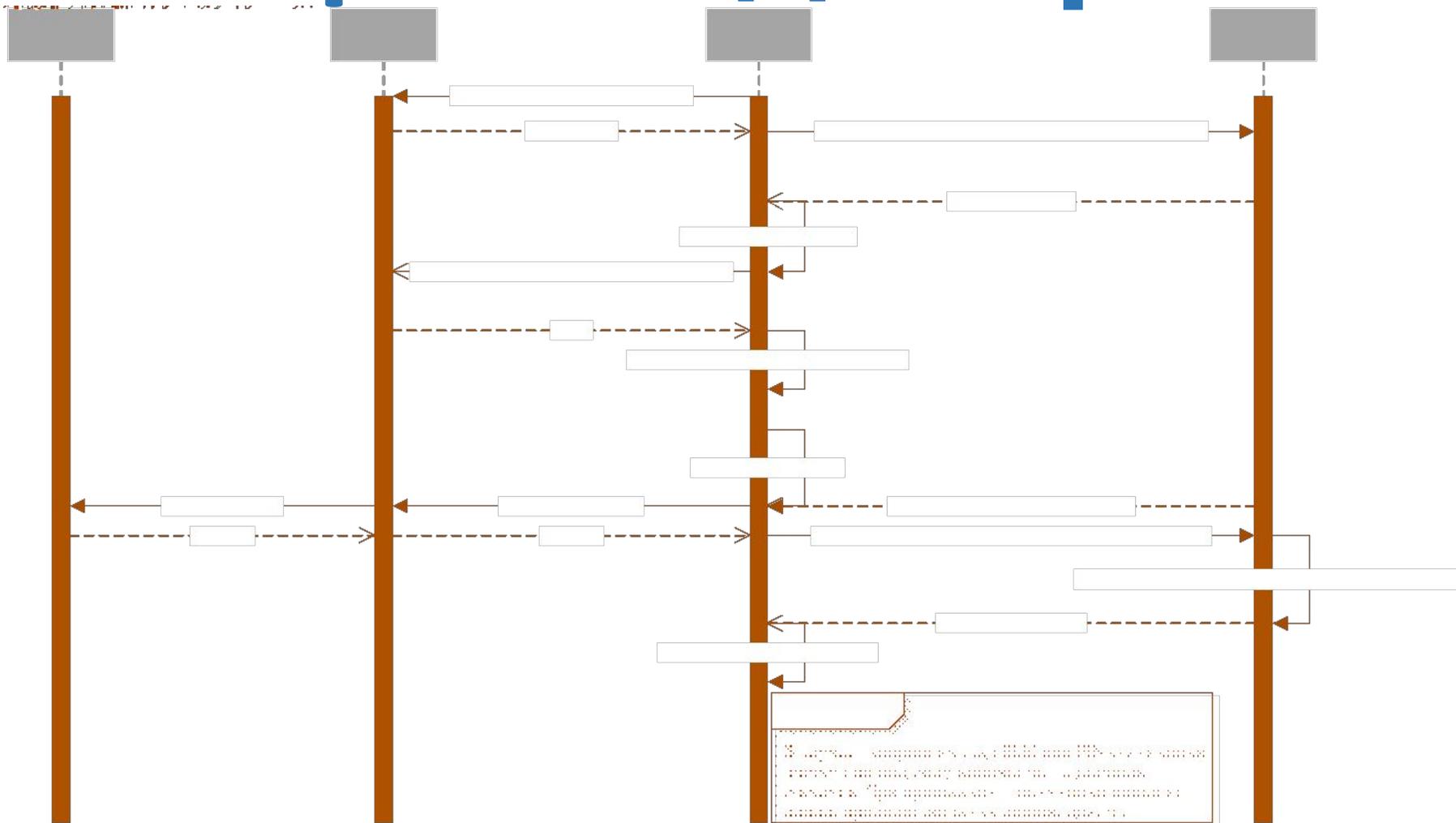
Сообщение самому себе



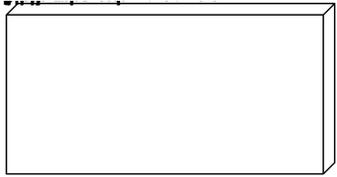
Фрагмент цикла. В блок помещаются элементы диаграммы, требующие повторения по заданным условиям



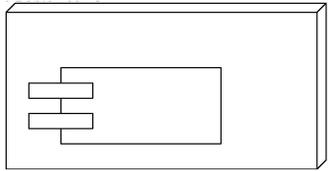
Sequence диаграмма



Deployment диаграмма



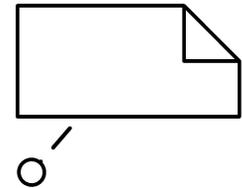
Узел системы, представляет собой аппаратный элемент



Узел с компонентом системы. Указываются только исполняемые компоненты



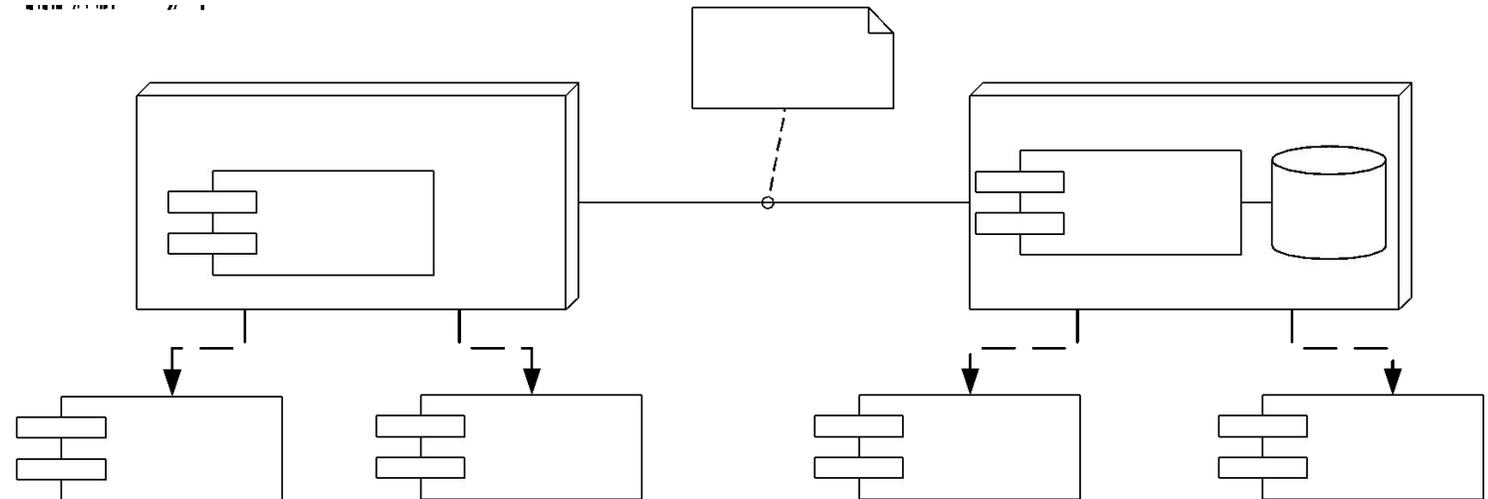
Отношения между узлами системы



Комментарии



Зависимости между узлами и компонентами





Activity диаграмма

 Начальный узел, показывает начало процесса

 Конечный узел, показывает окончание процесса
 Узел ветвления, он же является узлом слияния ветвления. Работает по принципу ИЛИ

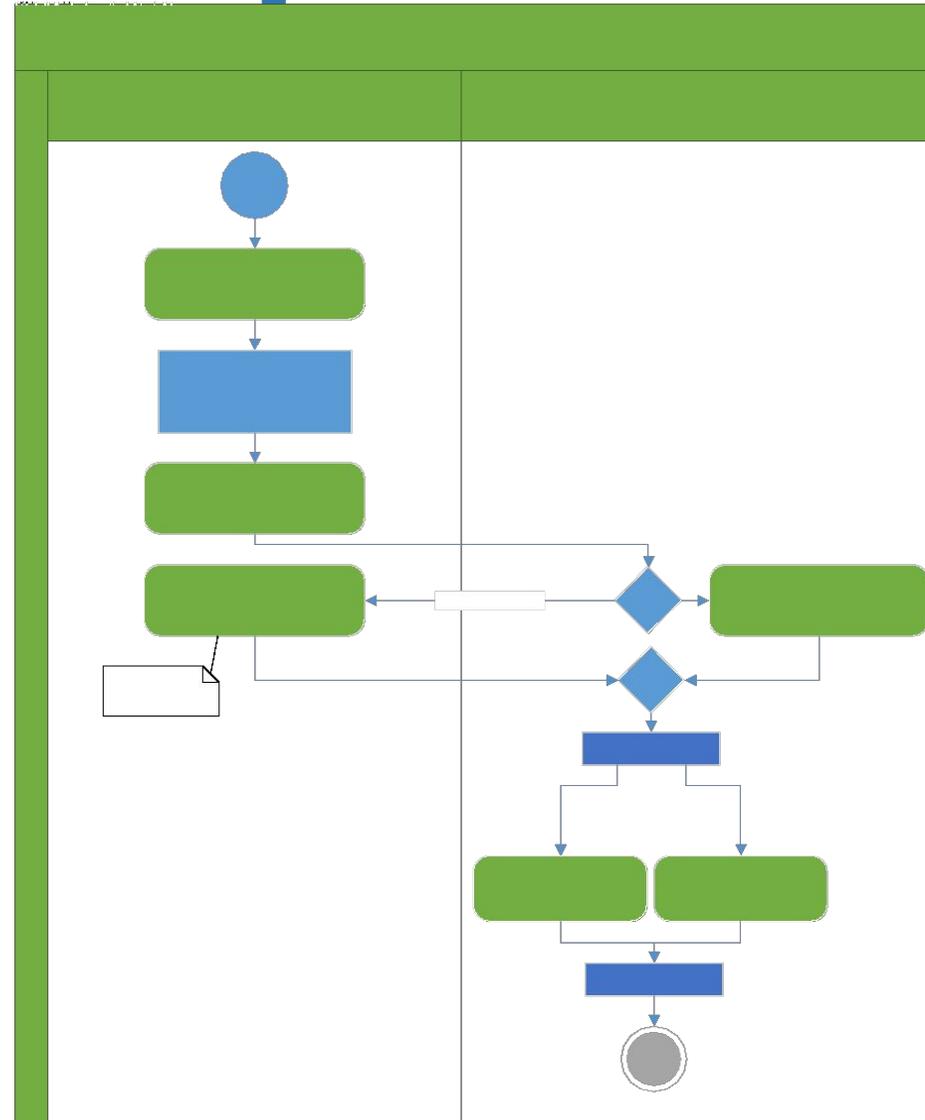
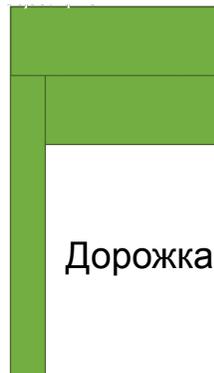
 Узел ветвления, он же является узлом слияния ветвления. Работает по принципу И. Используется для параллельных процессов

 Действие

 Комментарий

 Объект

 Переходы



Литература

<https://studopedia.info/5-96298.html> - жизненный цикл разработки ПО

<http://docs.cntd.ru/document/gost-34-601-90> - ГОСТ 34.601-90 этапы разработки ПО

<https://analytics.infozone.pro/requirements-analysis/analysis-of-requirements-wiegers-2004/> - этапы сбора требований

<https://www.webursitet.ru/article/vidy-trebovanii-k-programmnomu-produktu.html> - виды требований с примерами

<https://docs.google.com/file/d/0BxOg0amRzk9vbzdPcUFobzA5aWs/edit?usp=sharing> – книга Карла Вигерса

<https://habr.com/ru/company/edison/blog/269789/> - основные методологии разработки ПО

<https://worksection.com/blog/waterfall-vs-agile.html> - сравнение WaterFall с Agile

<https://ru.wikipedia.org/wiki/UML> - виды UML диаграмм

<http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html> - диаграмма вариантов использования

<http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl8/gl8.html> - диаграмма последовательности

<http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl11/gl11.html> - диаграмма развертывания

<http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html> - диаграмма действий

Задание

Заказчик сформировал требования по добавлению на их сайте on-line кредитного калькулятора с возможностью формирования заявки на кредит через сайт.

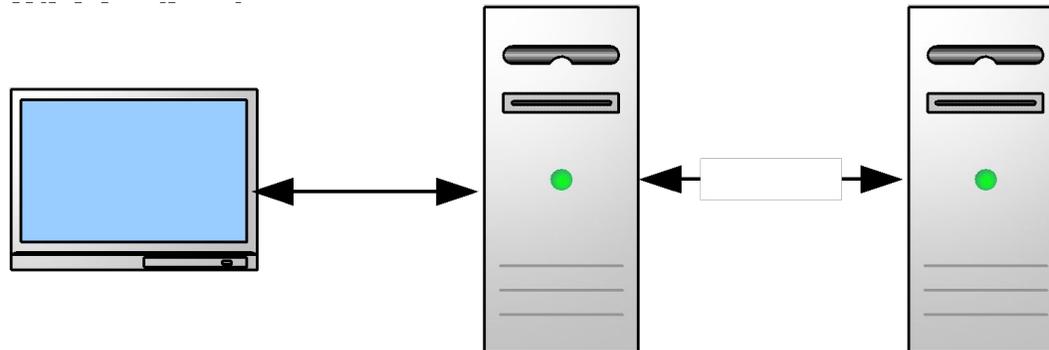
Текущая архитектура Банка состоит из 3-х основных узлов: АБС (хранит информацию о клиентах банка, ставках по кредитам, максимальной сумме кредита и его сроке), Web сервер отвечает за формирование Web страницы и сама web страница.

От заказчика поступили следующие требования:

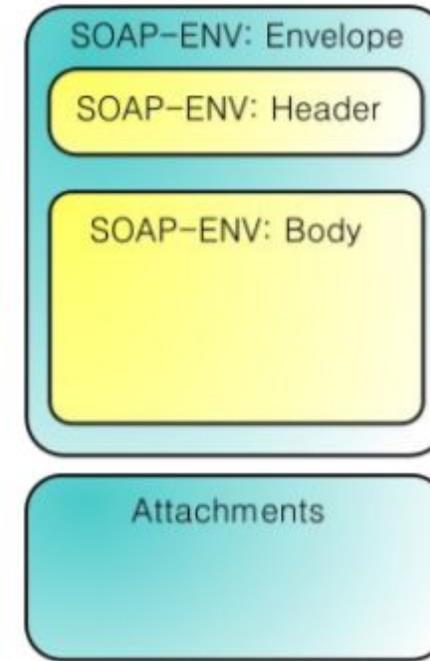
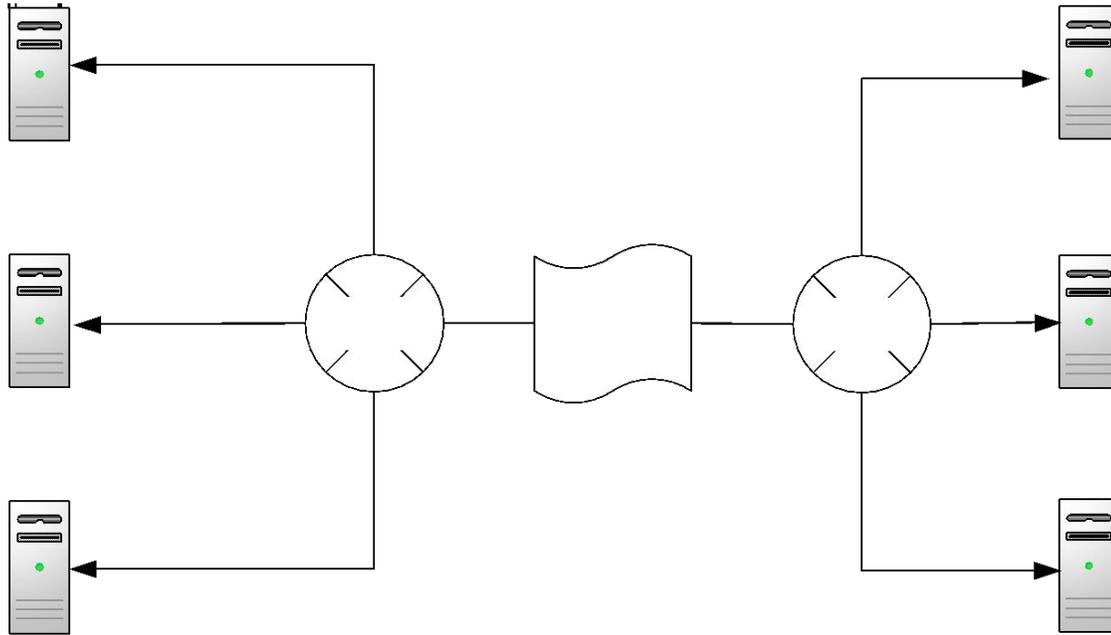
1. На страницу банка необходимо добавить новый раздел on-line калькулятор;
2. Максимальная сумма кредита, максимальный срок и ставка хранятся в АБС. Требуется передавать эти значения из АБС на Web Server и отображать клиенту. Текущая реализация не содержит интеграции между АБС и Web сервером;
3. Клиенту на странице on-line калькулятора доступны: выбор суммы кредита, выбор срока кредита;
4. Выбрав указанные значения происходит перерасчет суммы ежемесячного платежа. Выбор места расчета платежа на web сервере или в браузере остается на усмотрения исполнителя с объяснением причин выбора;
5. Взаимодействие между АБС и Web сервером – SOAP(xml), взаимодействие между Web-сервером и браузером HTTP(json);
6. Должна быть опция скачать в csv формате график платежей;
7. При выборе опции Оформить заявку, заполняются следующие реквизиты и отправляются в АБС: ФИО, дата рождения, телефон, email и выбранные условия (сумма, срок).

Задание

1. Описать Бизнес требования к задаче.
2. Описать варианты использования online кредитного калькулятора с использованием Use Case диаграммы
3. Сделать Sequence диаграмму взаимодействия между пользователем online калькулятора, Web сервером и АБС.
4. Сделать Deployment диаграмму разрабатываемого решения
5. С помощью Activity диаграммы описать функциональную архитектуру системы



SOAP – Simple Object Access Protocol



SOAP-ENV: Envelope

<SOAP-ENV:Envelope
xmlns:SOAP-ENV=<http://www.w3.org/2003/05/soap-envelope>> для версии 1.2

<SOAP-ENV:Envelope
xmlns:SOAP-ENV=<http://schemas.xmlsoap.org/soap/>>
для версии 1.1

SOAP – Simple Object Access Protocol

encodingStyle – идентификатор пространства имен для определенных типов в XSD

actor (или **role** для версии 1.2) - Тип данных URI. Задает адрес конкретного SOAP-сервера, которому предназначено сообщение.

mustUnderstand – обязательность учитывать синтаксис элемента при его

обработке

relay – показывает, передавать или удалять дальше заголовочный блок при передаче сообщения от текущего узла в следующий

Версия 1.1

faultcode – код ошибки

faultstring – описание ошибки

faultactor – URI адрес сервера обнаружившего ошибку

detail – детальное описание ошибки

Версия 1.2

code – код ошибки

reason – причина ошибки

node - URI адрес сервера обнаружившего ошибку

role – роль узла, обнаружившего ошибку

detail – детальное описание ошибки

SOAP-ENV: Header

SOAP-ENV: Body

SOAP – Simple Object Access Protocol

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <env:Fault>
      <faultcode>env:MustUnderstand</faultcode>
      <faultstring>SOAP Must Understand Error</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" xmlns:rpc='http://www.w3.org/2002/06/soap-rpc1'>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>Processing Error</env:Reason>
      <env:Detail>
        <e:myfaultdetails xmlns:e="http://www.example.org/faults">
          <message>Name does not match</message>
          <errorcode>999</errorcode>
        </e:myfaultdetails>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

SOAP – Simple Object Access Protocol

Запрос

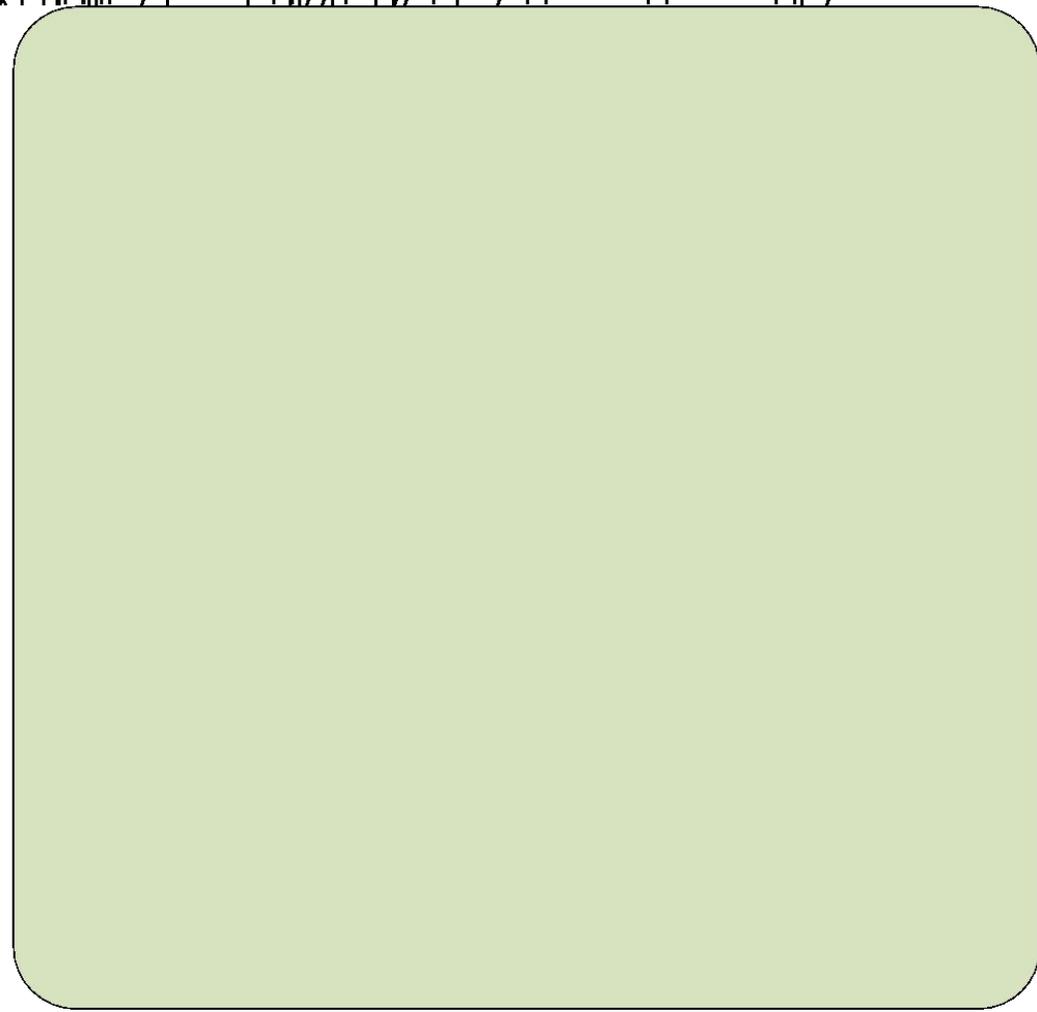
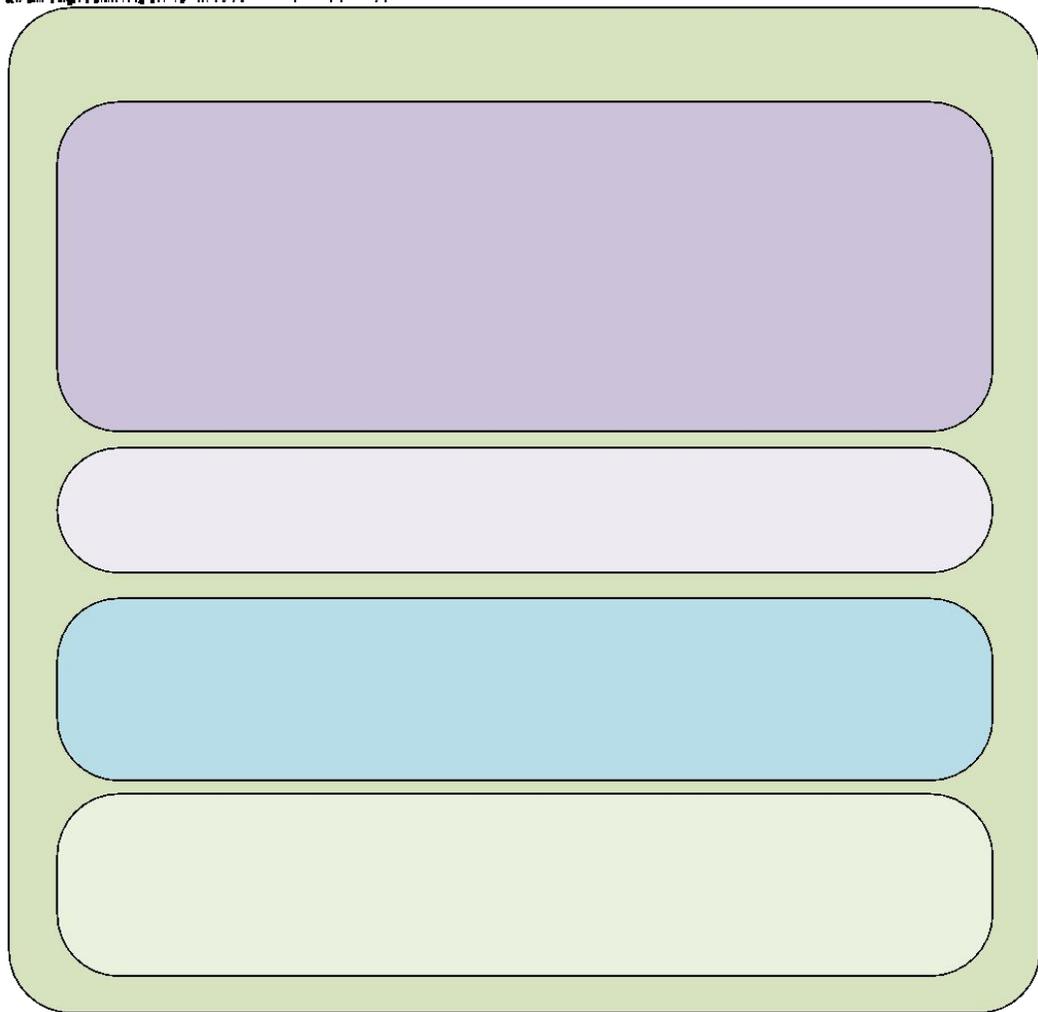
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <authorizeRequest xmlns="http://warehouse.example.com/ws">
      <login>12345</login>
      <password>12345</password>
    </authorizeRequest>
  </soap:Body>
</soap:Envelope>
</env:Envelope>
```

Ответ

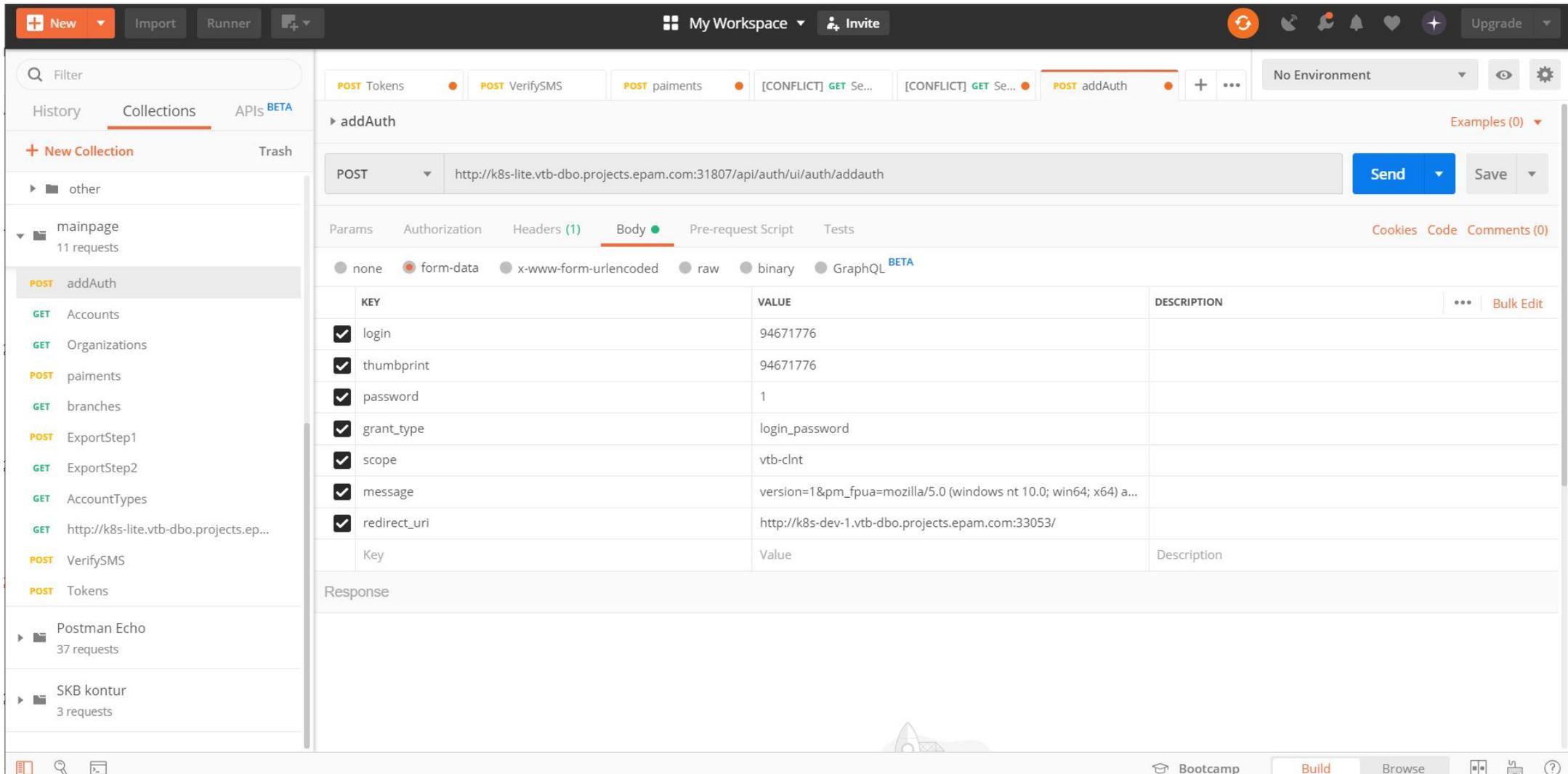
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AuthorizeResponse xmlns="http://warehouse.example.com/ws">
      <AuthorizationInformation>
        <login>12345</login>
        <person>
          <Name>Name</Name>
          <Surname>Surname</Surname>
        </person>
        <accessToken>fd3rfdsf42</AccessToken>
      </AuthorizationInformation>
    </AuthorizeResponse>
  </soap:Body>
</soap:Envelope>
```



HTTP – HyperText Transfer Protocol



HTTP – HyperText Transfer Protocol

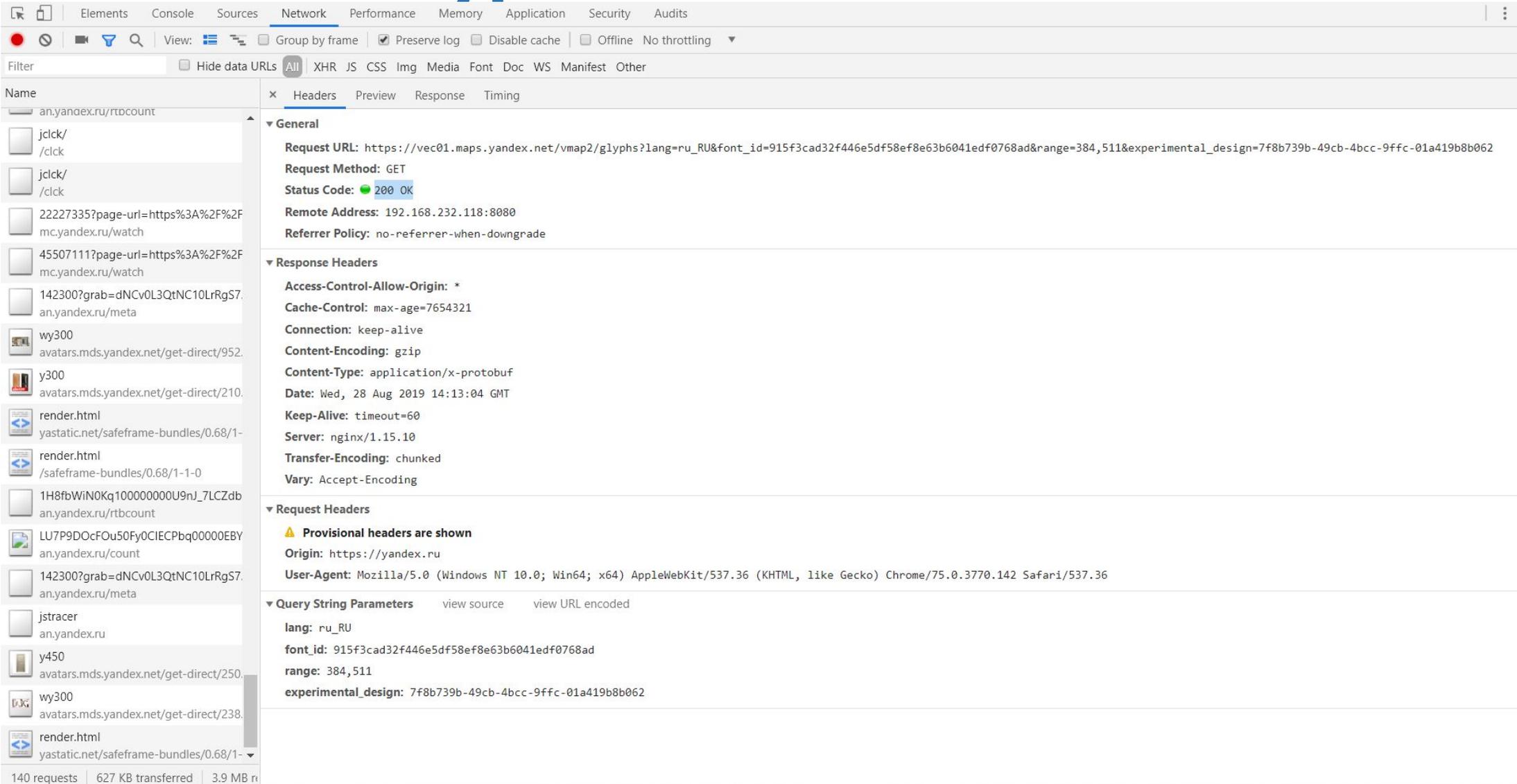


The screenshot displays the Postman interface for configuring a REST client request. The request is a POST to the endpoint `http://k8s-lite.vtb-dbo.projects.epam.com:31807/api/auth/ui/auth/addauth`. The request body is configured as form-data with the following parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> login	94671776	
<input checked="" type="checkbox"/> thumbprint	94671776	
<input checked="" type="checkbox"/> password	1	
<input checked="" type="checkbox"/> grant_type	login_password	
<input checked="" type="checkbox"/> scope	vtb-clnt	
<input checked="" type="checkbox"/> message	version=1&pm_fpu=mozilla/5.0 (windows nt 10.0; win64; x64) a...	
<input checked="" type="checkbox"/> redirect_uri	http://k8s-dev-1.vtb-dbo.projects.epam.com:33053/	
Key	Value	Description

The interface also shows a sidebar with a collection of requests, including 'addAuth', 'Accounts', 'Organizations', 'paiments', 'branches', 'ExportStep1', 'ExportStep2', 'AccountTypes', 'VerifySMS', and 'Tokens'. The bottom status bar includes 'Bootcamp', 'Build', 'Browse', and other utility icons.

HTTP – HyperText Transfer Protocol



The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The request is for the URL `https://vec01.maps.yandex.net/vmap2/glyphs?lang=ru_RU&font_id=915f3cad32f446e5df58ef8e63b6041edf0768ad&range=384,511&experimental_design=7f8b739b-49cb-4bcc-9ffc-01a419b8b062`. The status is 200 OK. The response headers include `Access-Control-Allow-Origin: *`, `Cache-Control: max-age=7654321`, `Content-Encoding: gzip`, and `Content-Type: application/x-protobuf`. The request headers show the origin as `https://yandex.ru` and the user agent as `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36`. The query string parameters are `lang: ru_RU`, `font_id: 915f3cad32f446e5df58ef8e63b6041edf0768ad`, `range: 384,511`, and `experimental_design: 7f8b739b-49cb-4bcc-9ffc-01a419b8b062`.

Elements Console Sources Network Performance Memory Application Security Audits

View: [Icons] Group by frame [x] Preserve log [x] Disable cache [x] Offline No throttling [v]

Filter [x] Hide data URLs [x] All XHR JS CSS Img Media Font Doc WS Manifest Other

Name [x] Headers Preview Response Timing

an.yandex.ru/rtbcount

- jcick/
/clck
- jcick/
/clck
- 22227335?page-url=https%3A%2F%2Fmc.yandex.ru/watch
- 45507111?page-url=https%3A%2F%2Fmc.yandex.ru/watch
- 142300?grab=dNCv0L3QtNC10LrRgS7.
an.yandex.ru/meta
- wy300
avatars.mds.yandex.net/get-direct/952.
- y300
avatars.mds.yandex.net/get-direct/210.
- render.html
yastatic.net/safeframe-bundles/0.68/1-
- render.html
/safeframe-bundles/0.68/1-1-0
- 1H8fbWiN0Kq10000000U9nJ_7LCZdb
an.yandex.ru/rtbcount
- LU7P9DOcFOu50Fy0CIECPbq0000EBY
an.yandex.ru/count
- 142300?grab=dNCv0L3QtNC10LrRgS7.
an.yandex.ru/meta
- jstracer
an.yandex.ru
- y450
avatars.mds.yandex.net/get-direct/250.
- wy300
avatars.mds.yandex.net/get-direct/238.
- render.html
yastatic.net/safeframe-bundles/0.68/1-

140 requests | 627 KB transferred | 3.9 MB r

General

Request URL: `https://vec01.maps.yandex.net/vmap2/glyphs?lang=ru_RU&font_id=915f3cad32f446e5df58ef8e63b6041edf0768ad&range=384,511&experimental_design=7f8b739b-49cb-4bcc-9ffc-01a419b8b062`

Request Method: GET

Status Code: 200 OK

Remote Address: 192.168.232.118:8080

Referrer Policy: no-referrer-when-downgrade

Response Headers

Access-Control-Allow-Origin: *

Cache-Control: max-age=7654321

Connection: keep-alive

Content-Encoding: gzip

Content-Type: application/x-protobuf

Date: Wed, 28 Aug 2019 14:13:04 GMT

Keep-Alive: timeout=60

Server: nginx/1.15.10

Transfer-Encoding: chunked

Vary: Accept-Encoding

Request Headers

Provisional headers are shown

Origin: `https://yandex.ru`

User-Agent: `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36`

Query String Parameters [view source](#) [view URL encoded](#)

lang: `ru_RU`

font_id: `915f3cad32f446e5df58ef8e63b6041edf0768ad`

range: `384,511`

experimental_design: `7f8b739b-49cb-4bcc-9ffc-01a419b8b062`

RestFull Api

Модель взаимодействия клиент-сервер (Client-Server) – взаимодействие происходит по принципу клиент серверной архитектуры

Отсутствие состояния (Stateless) - Сервер не должен хранить какой-либо информации о клиентах. Вся необходимая информация о клиенте хранится в запросах.

Кеширование (Cache) - клиенты могут кэшировать ответы. Каждый ответ должен быть отмечен является ли он кэшируемым или нет.

Единообразие интерфейса (Uniform Interface) – любой Rest сервис должен быть понятен без его разработчика.

Многоуровневая система (Layered System)- Клиент не может однозначно определить, подключается ли он непосредственно к серверу или к посреднику по пути подключения.

Код по требованию (Code-On-Demand) – опциональное условие. Сервер может отдавать клиенту исполняемый код (апплеты)

XML формат и XSD схема

В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация

Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника"

В XML учитывается регистр символов

Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки. Допускается несколько атрибутов в пределах одного тега. Атрибут указывается в начальном теге.

Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов

Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML)

XML формат и XSD схема

```
<?xml version="1.0" encoding="utf-8"?>
<users xsi:noNamespaceSchemaLocation="schema.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <UserInformation UserType="External">
    <Login>string</Login>
    <Password>string</Password>
    <Person>
      <Name>string</Name>
      <Surname>string</Surname>
    </Person>
  </UserInformation>
  <UserInformation UserType="Internal">
    <Login>string</Login>
    <Password>string</Password>
    <Person>
      <Name>string</Name>
      <Surname>string</Surname>
    </Person>
  </UserInformation>
</users>
```

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="users">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserInformation" type="UserInformationType"
minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="UserInformationType">
    <xs:sequence>
      <xs:element name="Login" type="xs:string" />
      <xs:element name="Password" type="xs:string" />
      <xs:element name="Person" type="PersonType" />
    </xs:sequence>
    <xs:attribute name="UserType" type="xs:string" />
  </xs:complexType>
  <xs:complexType name="PersonType">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" />
      <xs:element name="Surname" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

JSON формат и JSON Schema

JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript.

В качестве значений в JSON могут быть использованы:

Запись — это неупорядоченное множество пар **ключ:значение**, заключённое в фигурные скобки «`{ }`». Ключ описывается **строкой**, между ним и значением стоит символ «`:`». Пары *ключ-значение* отделяются друг от друга запятыми.

Массив (одномерный) — это упорядоченное множество **значений**. Массив заключается в квадратные скобки «`[]`». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения.

Число (целое или вещественное).

Литералы *true*, *false* и *null*.

Строка — это упорядоченное множество из нуля или более символов, заключённое в двойные кавычки.

JSON формат и JSON Schema

```
{
  "USERS": [
    {
      "UserInformation": {
        "isExternal": true,
        "Login": "12345",
        "Password": "12345",
        "Person": {
          "Surname": "Surname",
          "Name": "Name",
          "age": 32
        }
      }
    },
    {
      "UserInformation": {
        "isExternal": False,
        "Login": "12345",
        "Password": "12345",
        "Person": {
          "Surname": "Surname",
          "Name": "Name"
        }
      }
    }
  ]
}
```

Типа данных в JsonSchema

- string (строка)
- number (число, включая все действительные числа)
- integer (целое число, является подмножеством number)
- boolean (true или false)
- object (объект, в некоторых языках зовётся ассоциативным массивом, хэшем, хэш-таблицей, картой или словарём)
- array (массив)
- null («нет данных» или «не известно», возможно только значение null)
- any (любой тип, включая null)



JsonSchema.txt

JSON формат и JSON Schema

Типа данных в Json Schema

- **string** (строка)
- **number** (число, включая все действительные числа)
- **integer** (целое число, является подмножеством number)
- **boolean** (true или false)
- **object** (объект, в некоторых языках зовётся ассоциативным массивом, хэшем, хэш-таблицей, картой или словарём)
- **array** (массив)
- **null** («нет данных» или «не известно», возможно только значение null)
- **any** (любой тип, включая null)

CSV (Comma-Separated Values) формат

Спецификация:

- Каждая строка файла — это одна строка таблицы.
- Разделителем значений колонок является символ запятой (,). Однако на практике часто используются другие разделители.
- Значения, содержащие зарезервированные символы (двойная кавычка, запятая, точка с запятой, новая строка) обрамляются двойными кавычками ("). Если в значении встречаются кавычки — они представляются в файле в виде двух кавычек подряд.

```
isExternal;Login;Password;Surname;Name;age
```

```
true;12345;12345;Surname1;Name1;32
```

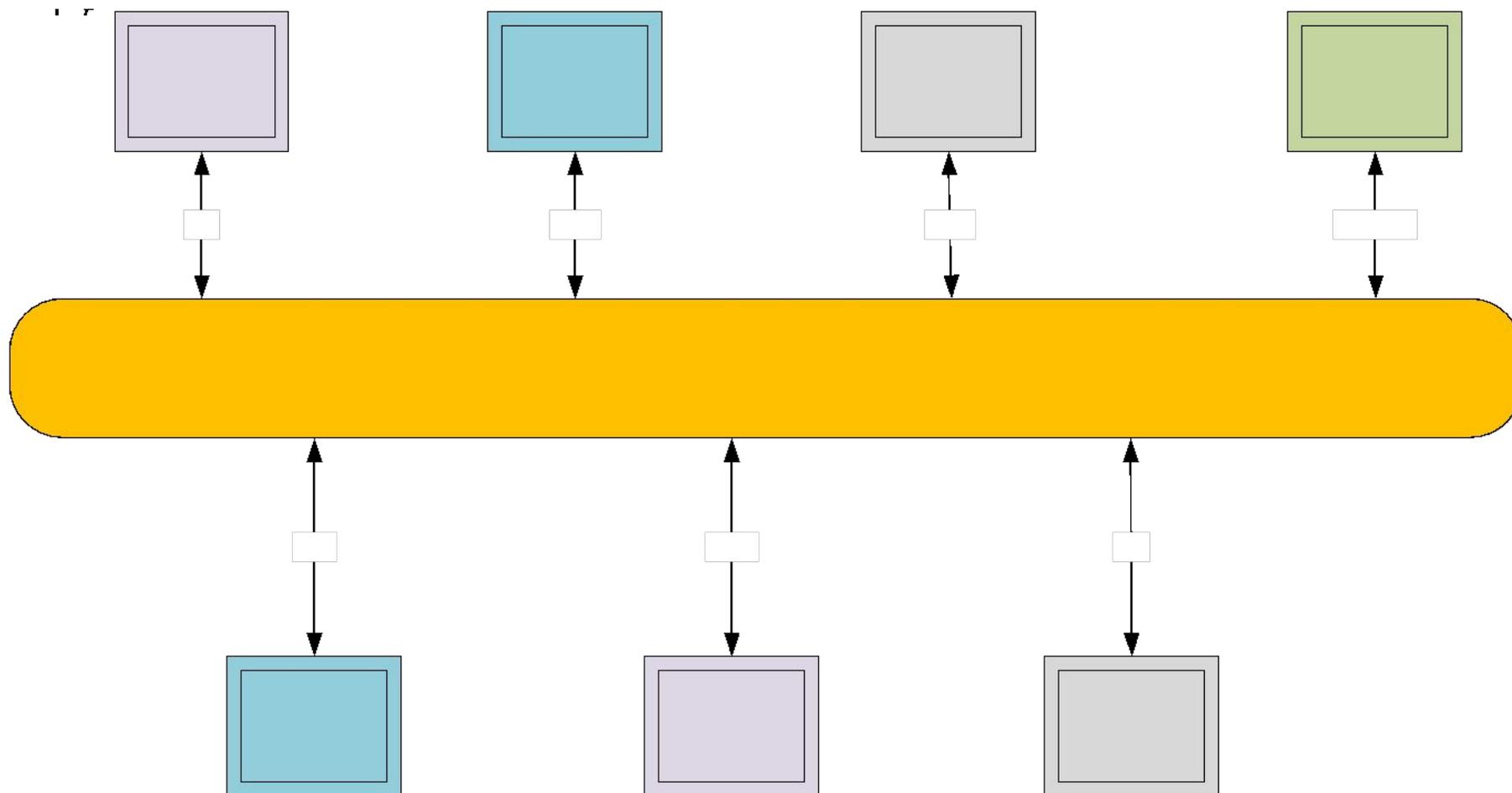
```
false;12345;12345;Surname2;Name2;
```



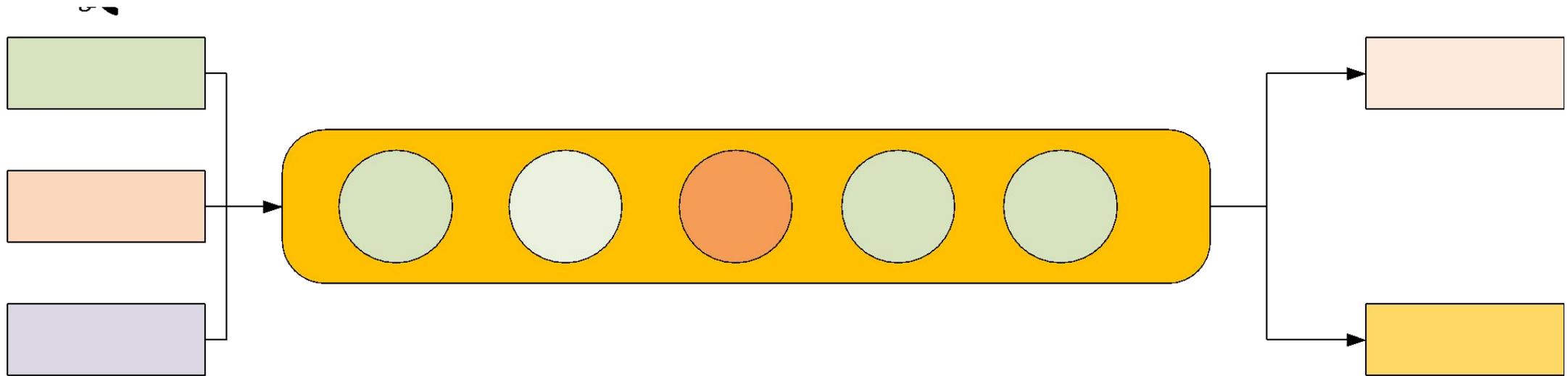
ТЕХНОСЕРВ
КОНСАЛТИНГ

ESB(Enterprise Service Bus)

ИНТЕГРАЦИЯ

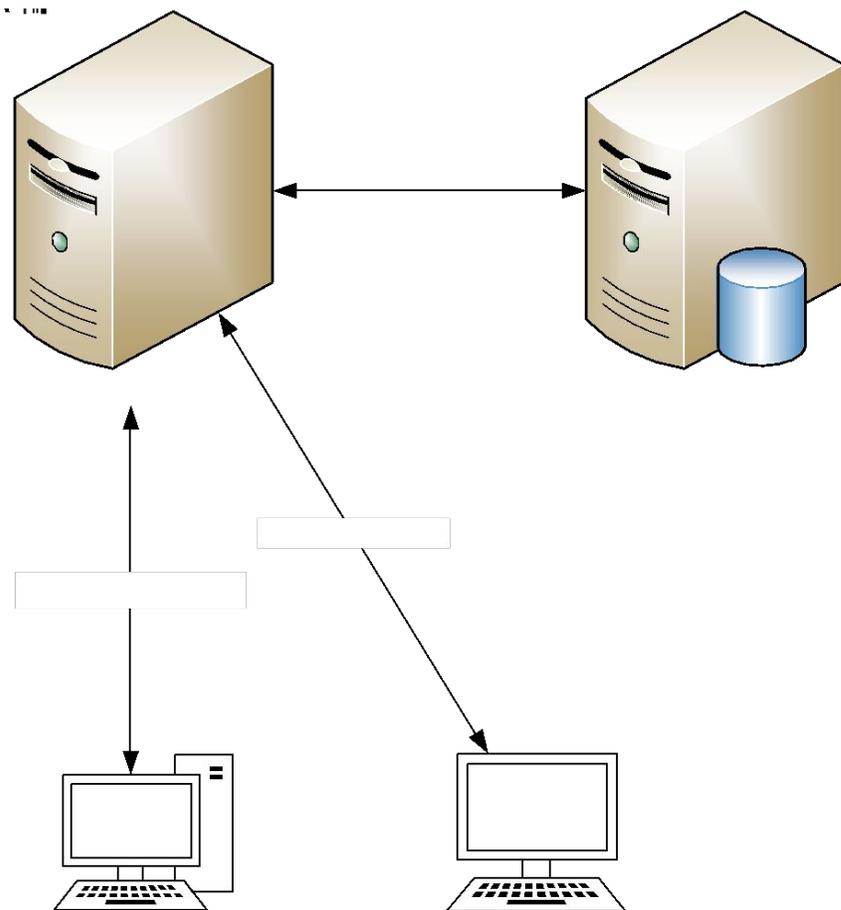


MQ (Message Queue) интеграция



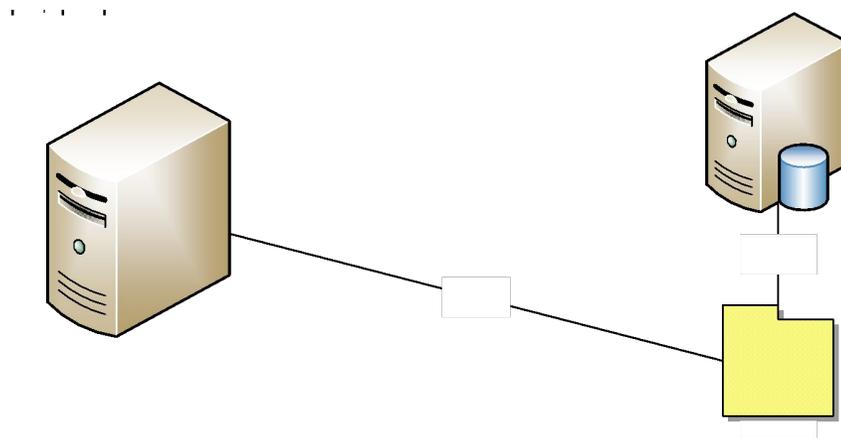
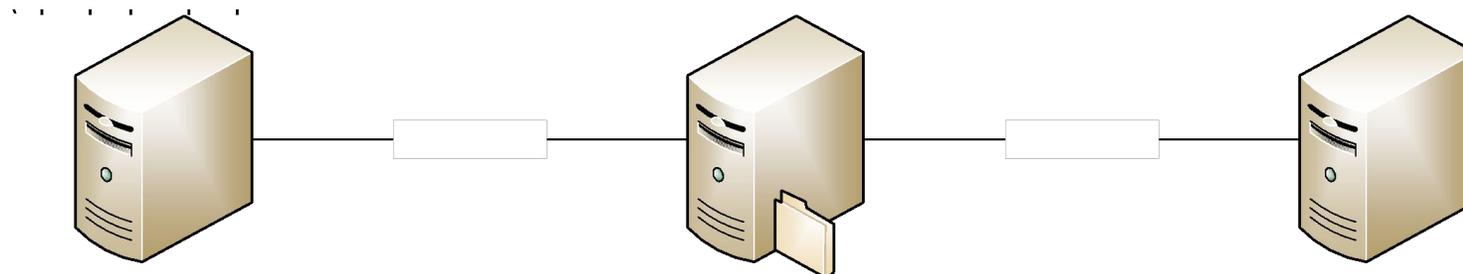


API w\s (Web Service) интеграция

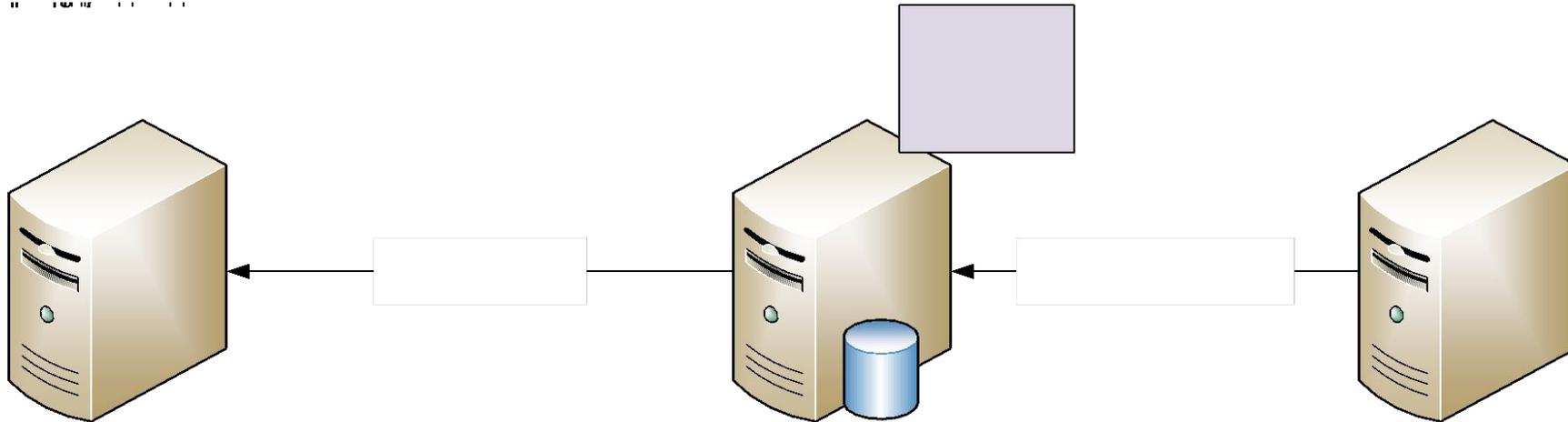




Интеграция через файл

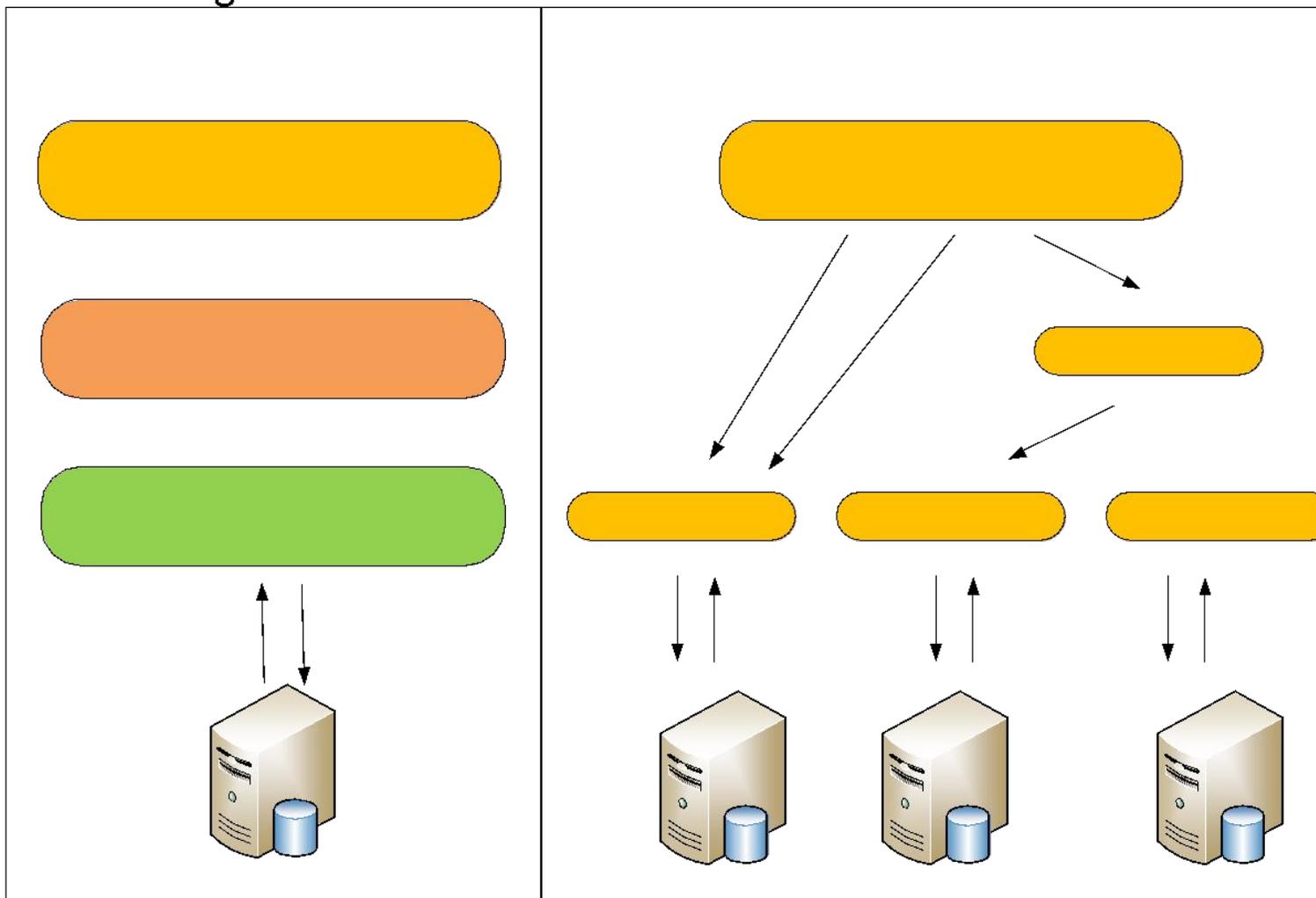


Интеграция через таблицу в БД





Микросервисная архитектура



Литература

<http://khpi-iip.mipk.kharkiv.edu/library/sotii/lectures/Lecture5.pdf> - SOAP протокол

<https://ru.wikipedia.org/wiki/HTTP#OPTIONS> – HTTP протокол

<https://habr.com/ru/post/319984/> - Rest системы

<http://sap.pitroff.ru/tehnologii/rest/rest-eto-ne-pro-otdyih-chast-pervaya-chto-takoe-rest/> - ЧТО ТАКОЕ REST архитектура

<http://blogger.sapronov.me/2014/02/rest.html> - ВВЕДЕНИЕ В rest

<https://www.ibm.com/developerworks/ru/library/x-newxml/index.html> - xml формат

<https://bdpx.github.io/xml/lab3/xsd.html> - xsd схема

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/linq/sample-xsd-file-customers-and-orders1> - образец XSD файла

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/linq/sample-xml-file-customers-and-orders-ling-to-xml-2> - образец XML файла

<https://ru.wikipedia.org/wiki/JSON> -JSON формат

<https://habr.com/ru/post/158927/> - JSON Schema

<https://ru.wikipedia.org/wiki/CSV> - CSV формат

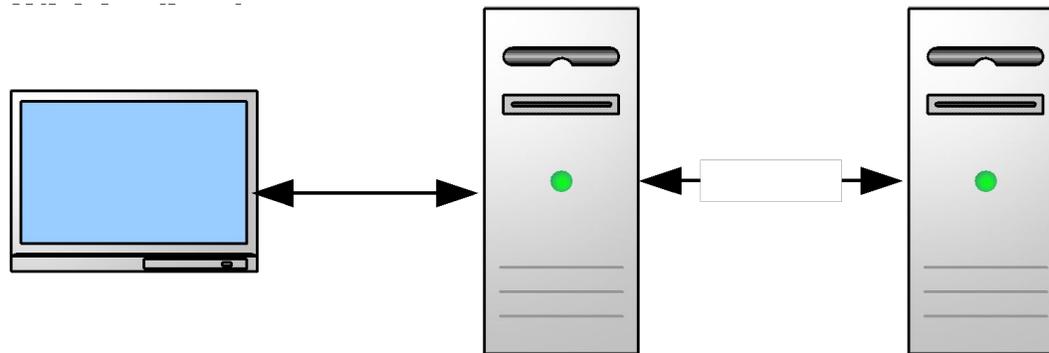
<https://compress.ru/article.aspx?id=21413> – ESB интеграция

<https://habr.com/ru/post/326088/> - виды интеграций

<https://javarush.ru/groups/posts/2015-mikroservisnaja-arhhitektura-pljusih-i-minusih> - микросервисная архитектура

Задание

1. Описать в формате JSON взаимодействие Web сервер – браузер
 - JSON с данными по условиям кредитования;
 - JSON с заявкой на кредит;
 - В зависимости от выбранного решения по способу расчета ежемесячного платежа описать дополнительные сообщения.
2. В формате XML описать взаимодействие Web сервер – JSON
 - Данные по условиям кредитования;
 - Сформированная заявка.
3. Сделать XSD и JSON Schema для описываемых сообщений
4. Описать пример и формат графика платежей в CSV
5. Описать структуру в БД для хранения заявки на кредит
6. Изобразить интеграционную схему с протоколами взаимодействия





Структура технического задания

1. Глоссарий
2. Введение
3. Концепция
4. Требования к ППО
 - 4.1. Общие положения
 - 4.2. Функциональные требования
 - 4.2.1. Роли пользователей
 - 4.2.2. Описание функциональных требований
 - 4.3. Описание технологического процесса обработки данных
 - 4.4. Описание построения архитектуры
 - 4.5. Нефункциональные требования
 - 4.5.1. Требования к надежности и производительности
 - 4.5.2. Требования к информационной безопасности
 - 4.5.3. Требования к аудиту
 - 4.5.4. Требования к программному и аппаратному окружению
 - 4.5.5. Требования к документации
- 5*. Методика испытаний

6. Приложения



Microsoft Word



SMART принципы

Буква	Принцип
S	Specific (Конкретность)
M	Measurable (Измеримость)
A	Attainable (Достижимость)
R	Relevant (Уместность)
T	Time-bound (Ограниченность во времени)



ТЕХНОСЕРВ
КОНСАЛТИНГ

Задание

1. Описать ТЗ согласно структуре