Выполнил Матвеев Андрей

# Вводные данные

Перед вами представлены скриншоты фронт-офисного приложения для оформления заявок на получение кредита для физических лиц.

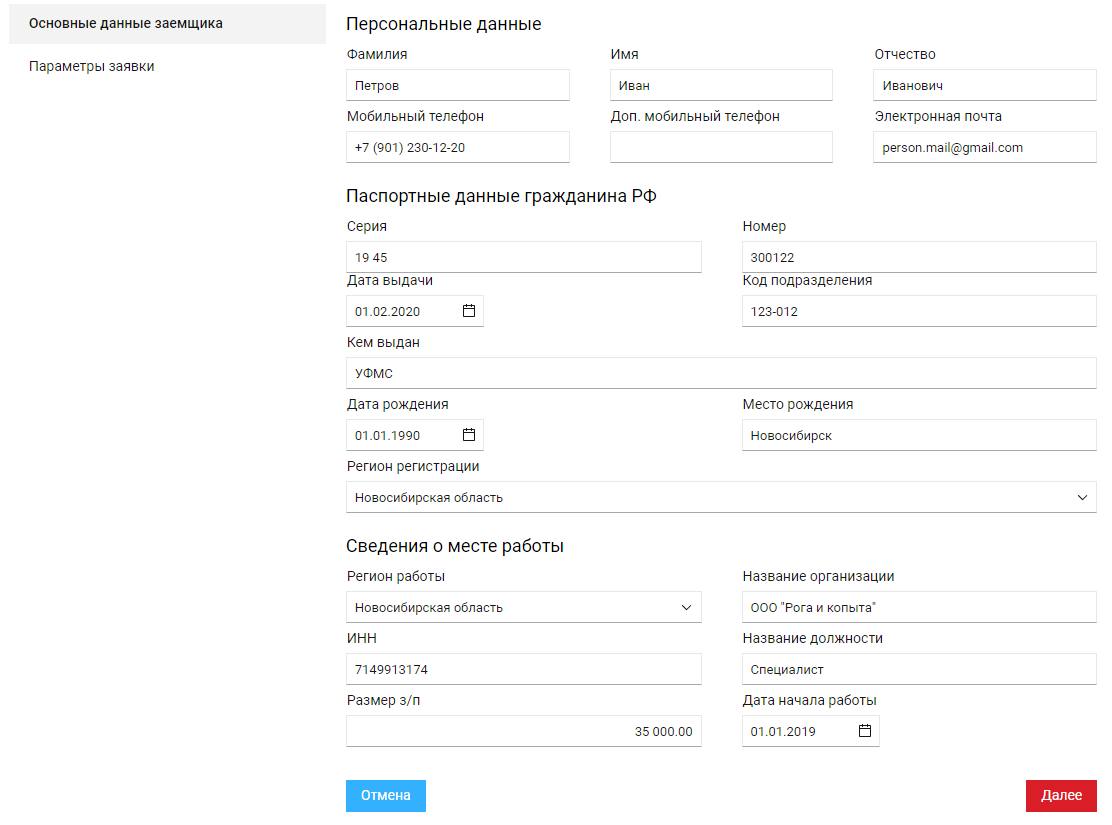


Рисунок 1

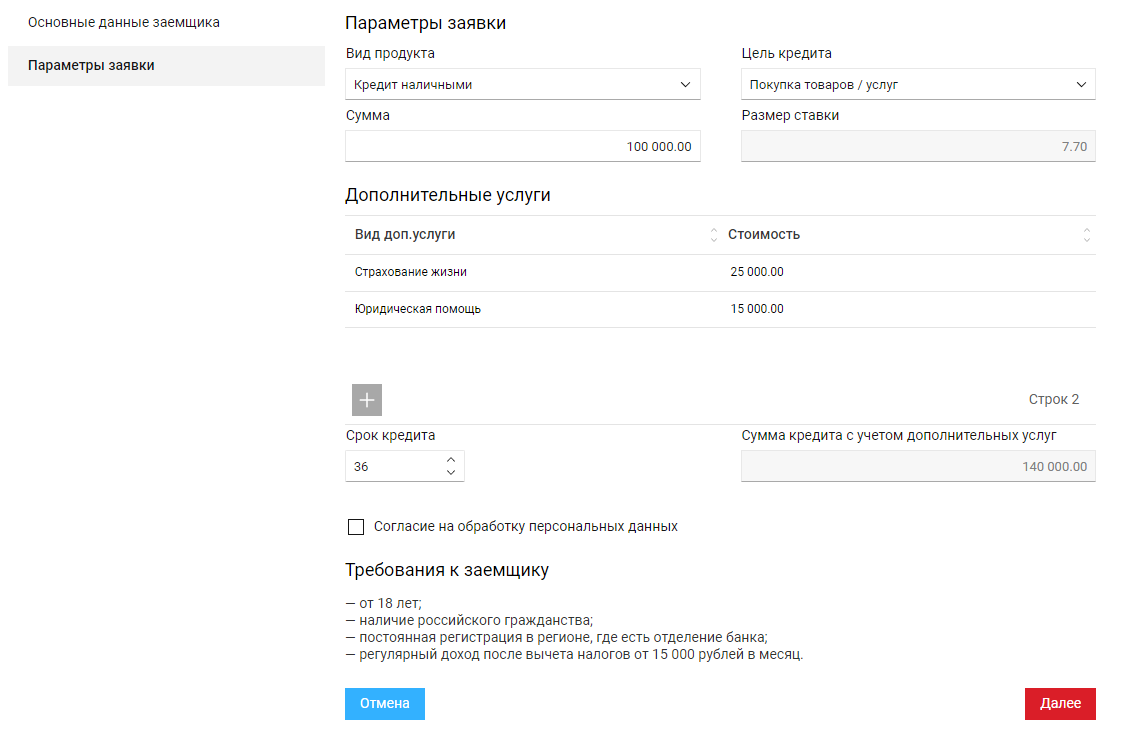


Рисунок 2

Ключевой функционал приложения: регистрация заявок на получение кредита, хранение информации о заемщиках и их заявках, кредитных продуктах и дополнительных услугах.

*Номера заданий лежат в соответствующих папках.*

*Папка 1 - задание 1. Представляет собой ER-диаграмму (ER.png) с с описанием сущностей и связей между таблицами.*

*База содержит большое количество таблиц для того, чтобы её было проще расширять и проще вносить изменения. Например, при изменении названия кредитного продукта для того, чтобы не изменять его во всех строках таблицы заявки на кредит, можно просто вынести в отдельную таблицу виды кредитных продуктов и связать с таблицей заявки на кредит при помощи внешнего ключа. В ER-модели прописано также обоснование связей между таблицами.*

*Также в папке '1' содержится спроектированная база данных (DB.png) с описанием связей между таблицами, с разделением внешних и внутренних ключей, промежуточными таблицами в случае связи "многие ко многим" и с описанием типов.*

*А еще, чтобы проверить на практике правильность модели, реализована спроектированная база данных*

*в субд mysql Ver 8.0.29 (WebApp.sql).*

*Папка 2 - задание 2. Представляет файл с несколькими запросами (queries.sql) к спроектированной базе. ПОСЛЕДНИЙ ЗАПРОС - ОТВЕТ НА ЗАДАНИЕ 2. Приведено несколько запросов, чтобы показать ход мыслей, который привел к написанию искомого запроса. СУБД mysql Ver 8.0.29.*

*Папка 3 - задание 3.* Представляет скрипт на языке javascript. Файл AnnuityPaymentAlgorithm.js

Описание заданий приведено ниже.

## Задание 1

Опишите модель данных, которая может лежать в основе данного приложения.

Ожидаемый результат: ER диаграмма с пояснениями о выборе сущностей и типов данных их атрибутов. Используйте реляционную СУБД.

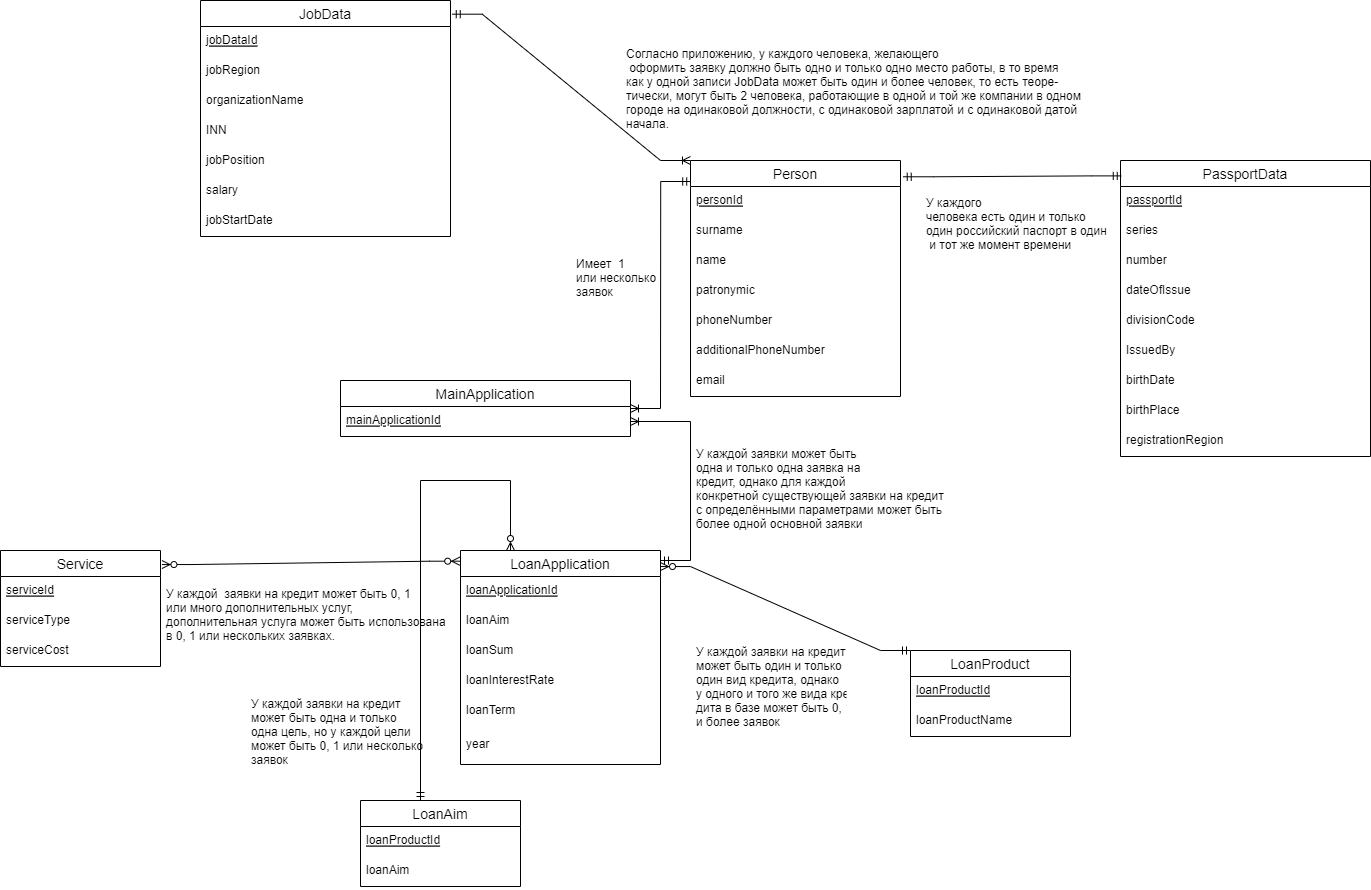


Рис. 1. ER-диаграмма.

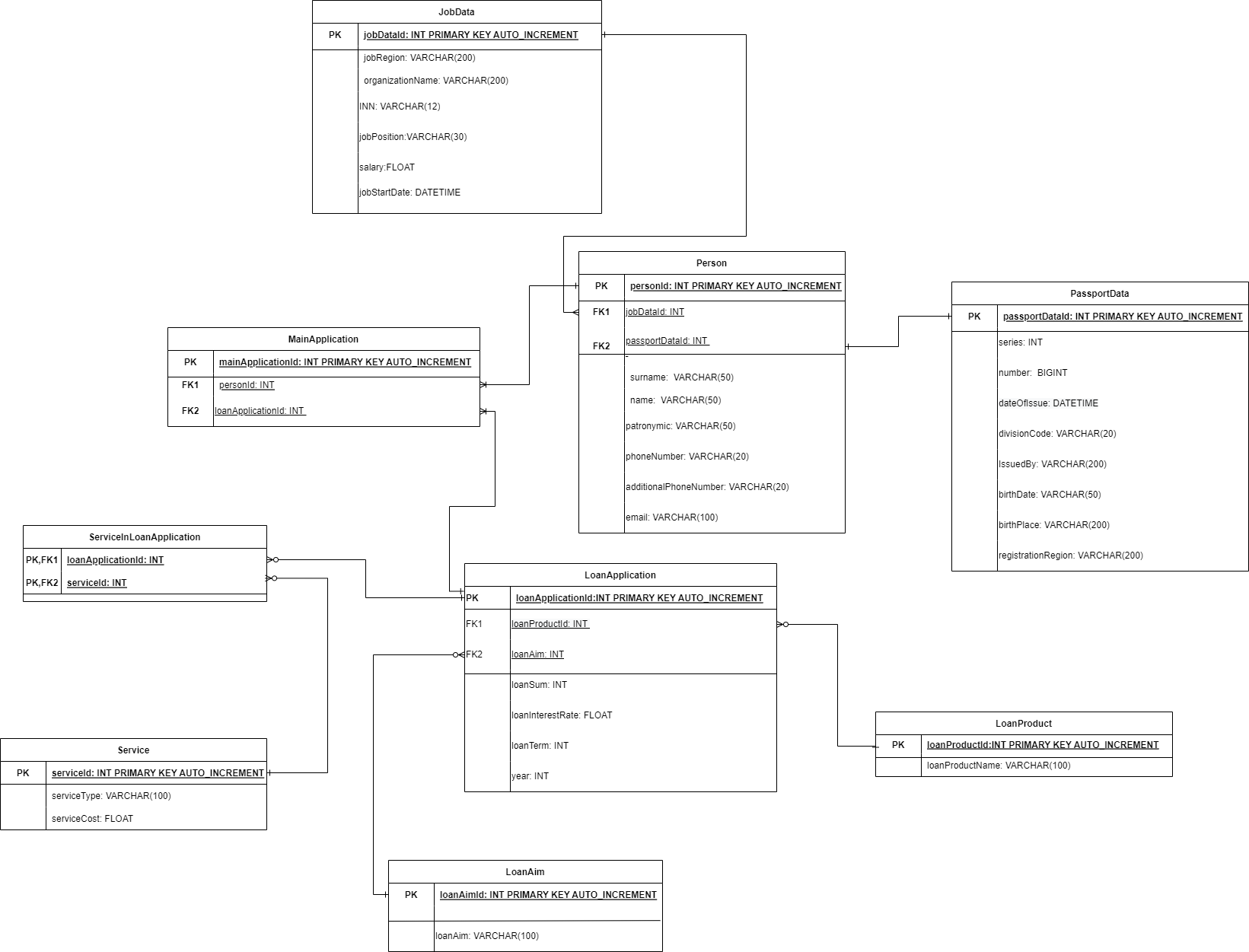


Рис. 2. Структура БД.

WebApp.sql

|  |
| --- |
| DROP DATABASE IF EXISTS WebApp;  CREATE DATABASE IF NOT EXISTS WebApp;  USE WebApp;  DROP TABLE IF EXISTS `MainApplication`;  DROP TABLE IF EXISTS `JobData`;  DROP TABLE IF EXISTS `Person`;  DROP TABLE IF EXISTS `LoanApplication`;  DROP TABLE IF EXISTS `LoanProduct`;  DROP TABLE IF EXISTS `ServiceInLoanApplication`;  DROP TABLE IF EXISTS `Service`;  DROP TABLE IF EXISTS `PassportData`;  DROP TABLE IF EXISTS `LoanAim`;  CREATE TABLE `LoanProduct`(  `loanProductId` INT PRIMARY KEY AUTO\_INCREMENT,  `loanProductName` VARCHAR(100) NOT NULL  );  CREATE TABLE `Service`(  `serviceId` INT PRIMARY KEY AUTO\_INCREMENT,  `serviceType` VARCHAR(100) NOT NULL,  `serviceCost` FLOAT NOT NULL  );  CREATE TABLE `LoanAim`(  `loanAimId` INT PRIMARY KEY AUTO\_INCREMENT,  `loanAim` VARCHAR(100) NOT NULL  );  CREATE TABLE `LoanApplication`(  `loanApplicationId` INT PRIMARY KEY AUTO\_INCREMENT,  `loanProductId` INT NOT NULL,  `loanAimId` INT NOT NULL,  `loanSum` FLOAT NOT NULL,  `loanInterestRate` FLOAT NOT NULL,  `loanTerm` INT NOT NULL,  `year` INT NOT NULL,  FOREIGN KEY (`loanProductId`) REFERENCES `LoanProduct`(`loanProductId`) ON DELETE CASCADE,  FOREIGN KEY (`loanAimId`) REFERENCES `LoanAim`(`loanAimId`) ON DELETE CASCADE  );  CREATE TABLE `ServiceInLoanApplication`(  `loanApplicationId` INT NOT NULL,  `serviceId` INT NOT NULL,  FOREIGN KEY (`loanApplicationId`) REFERENCES `LoanApplication`(`loanApplicationId`) ON DELETE CASCADE,  FOREIGN KEY (`serviceId`) REFERENCES `Service`(`serviceId`) ON DELETE CASCADE,  PRIMARY KEY(`loanApplicationId`, `serviceId`)  );  CREATE TABLE `PassportData`(  `passportDataId` INT PRIMARY KEY AUTO\_INCREMENT,  `series` INT NOT NULL,  `number` BIGINT NOT NULL,  `dateOfIssue` DATETIME NOT NULL,  `divisionCode` VARCHAR(20) NOT NULL,  `issuedBy` VARCHAR(200) NOT NULL,  `birthDate` VARCHAR(50) NOT NULL,  `birthPlace` VARCHAR(200) NOT NULL,  `registrationRegion` VARCHAR(200) NOT NULL  );  CREATE TABLE `JobData`(  `jobDataId` INT PRIMARY KEY AUTO\_INCREMENT,  `jobRegion` VARCHAR(200) NOT NULL,  `organizationName` VARCHAR(200) NOT NULL,  `INN` VARCHAR(12) NOT NULL,  `jobPosition` VARCHAR(30) NOT NULL,  `salary` FLOAT NOT NULL,  `jobStartDate` DATETIME NOT NULL  );  CREATE TABLE `Person`(  `personId` INT PRIMARY KEY AUTO\_INCREMENT,  `passportDataId` INT NOT NULL,  `jobDataId` INT NOT NULL,  `surname` VARCHAR(50) NOT NULL,  `name` VARCHAR(50) NOT NULL,  `patronymic` VARCHAR(50),  `phoneNumber` VARCHAR(20) NOT NULL,  `additionalPhoneNumber` VARCHAR(20),  `email` VARCHAR(100),  FOREIGN KEY (`passportDataId`) REFERENCES `PassportData`(`passportDataId`) ON DELETE CASCADE,  FOREIGN KEY (`jobDataId`) REFERENCES `JobData`(`jobDataId`) ON DELETE CASCADE  );  CREATE TABLE `MainApplication`(  `mainApplication` INT PRIMARY KEY AUTO\_INCREMENT,  `personId` INT NOT NULL,  `loanApplicationId` INT NOT NULL,  FOREIGN KEY (`personId`) REFERENCES `Person`(`personId`) ON DELETE CASCADE,  FOREIGN KEY (`loanApplicationId`) REFERENCES `LoanApplication`(`loanApplicationId`) ON DELETE CASCADE  );  INSERT INTO `LoanAim` (`loanAim`)  VALUES ('потребительский (товарный)'),  ('автокредит'),  ('ипотека'),  ('ссуда на получение образования'),  ('заем на лечение');  INSERT INTO `Service` (`serviceType`, `serviceCost`)  VALUES ('Страхование жизни', 25000.00),  ('Юридическая помощь', 15000.00);  INSERT INTO `LoanProduct` (`loanProductName`)  VALUES ('Кредит наличными'),  ('Кредит на карту'),  ('Кредит на текущий счет в банке');  INSERT INTO `LoanApplication` (`loanProductId`, `loanAimId`, `loanSum`, `loanInterestRate`, `loanTerm`, `year`)  VALUES (1, 2, 45000.00, 7.5, 12, 2022),  (2, 3,55000.00, 8.0, 36, 2021),  (1, 2,30000.00, 6.2, 15, 2022),  (2, 2,20000.00, 4.0, 14, 2022),  (3, 1, 1000000.00, 1.2, 18, 2021),  (1,4, 75000.00, 5.2, 50, 2022),  (3, 3,90000.00, 7.9, 30, 2022);  INSERT INTO `ServiceInLoanApplication`(`loanApplicationId`,`serviceId`)  VALUES (1, 1),  (2, 1),  (3, 1),  (4, 1),  (1, 2),  (2, 2),  (3, 2),  (4, 2);  INSERT INTO `JobData`(`jobRegion`, `organizationName`, `INN`, `jobPosition`, `salary`, `jobStartDate`)  VALUES ('Республика северная Осетия - Апания', 'Организация', '540256190', 'Слесарь', 45000.00 ,'1993-04-05 10:05:00' ),  ('Сахалинская область', 'Организация', '283700460', 'Сварщик', 60000.00 ,'1993-04-05 10:05:00'),  ('Ульяновская область', 'Организация', '455778262', 'Бухгалтер',60000.23,'1993-04-05 10:05:00'),  ('Смоленская область', 'Организация', '599011235','ДИректор', 42000.00,'1993-04-05 10:05:00'),  ('Томская область', 'Организация', '599011235', 'Повар', 75000.00 , '1993-04-05 10:05:00'),  ('Забайкальский край', 'Организация', '896644704', 'Электрик', 18000.00 ,'1993-04-05 10:05:00');  INSERT INTO `PassportData`(`series`, `number`, `dateOfIssue`, `divisionCode`, `issuedBy`, `birthDate`, `birthPlace`, `registrationRegion`)  VALUES (4393, 465970, '18.06.2019', '770-394', 'Отделом внутренних дел России по г. Пушкино', '08.05.1982', 'Россия, г. Пушкино, Юбилейная ул., д. 21 кв.45', 'Россия, г. Пушкино'),  (4854, 869961, '08.06.2016', '960-243', 'ОВД России по г. Элиста', '02.12.1993', 'Россия, г. Элиста, Вокзальная ул., д. 3 кв.205', 'Россия, г. Элиста'),  (4341, 577404, '13.08.2015', '800-452', 'Отделением УФМС России по г. Барнаул', '06.11.1961', 'г. Улан-Удэ', 'Россия, г. Улан-Удэ, Совхозная ул., д. 24 кв.30' ),  (4261, 819689, '18.06.2019', '770-394', 'Отделом внутренних дел России по г. Пушкино', '08.05.1982', 'Россия, г. Пушкино, Юбилейная ул., д. 21 кв.45', 'Россия, г. Пушкино'),  (4424, 329526, '08.06.2016', '960-243', 'ОВД России по г. Элиста', '02.12.1993', 'Россия, г. Элиста, Вокзальная ул., д. 3 кв.205', 'Россия, г. Элиста'),  (4142, 154853, '13.08.2015', '800-452', 'Отделением УФМС России по г. Барнаул', '06.11.1961', 'г. Улан-Удэ', 'Россия, г. Улан-Удэ, Совхозная ул., д. 24 кв.30');  INSERT INTO `Person`(`jobDataId`, `passportDataId`, `surname`, `name`, `patronymic`, `phoneNumber`, `additionalPhoneNumber`, `email`)  VALUES (1, 6, 'Николюк', 'Геннадий', 'Никифорович', '+7 (964) 784-30-86', '+7 (964) 784-30-86', 'gennadiy4432@outlook.com'),  (2, 5, 'Минеева', 'Нина', 'Феодосьевна' ,'+7 (964) 784-30-86', '+7 (964) 784-30-86', 'gennadiy4432@outlook.com'),  (3, 4, 'Шастин', 'Прохор', 'Петрович' ,'+7 (964) 784-30-86', '+7 (964) 784-30-86', 'gennadiy4432@outlook.com'),  (4, 3, 'Морякова', 'Таисия', 'Федоровна' ,'+7 (964) 784-30-86', '+7 (964) 784-30-86', 'gennadiy4432@outlook.com'),  (5, 2, 'Цыцын', 'Игнатий', 'Денисович' ,'+7 (964) 784-30-86', '+7 (964) 784-30-86', 'gennadiy4432@outlook.com'),  (6, 1, 'Андропова', 'Василиса', 'Тарасовна' ,'+7 (964) 784-30-86', '+7 (964) 784-30-86', 'vasilisa1973@mail.ru');  INSERT INTO `MainApplication`(`personId`, `loanApplicationId`)  VALUES (1, 1),  (1, 2),  (2, 3),  (3, 4),  (4, 5),  (5, 6),  (6, 7); |

## Задание 2

Напишите SQL-запрос, который бы возвращал самый популярный вид продукта за текущий год.

Ожидаемый результат: SQL-запрос, который основывается на модели данных из задания 1, и название СУБД, для которой был написан SQL-запрос. *СУБД mysql Ver 8.0.29.*

queries.sql

|  |
| --- |
| USE WebApp;  # для начала выведем фамилию, имя и название услуги, год взятия займа  SELECT `surname`, `name`, `loanProductName`, `year` FROM `MainApplication`  INNER JOIN `Person` USING(`personId`)  INNER JOIN `LoanApplication` USING(`loanApplicationId`)  INNER JOIN `LoanProduct` USING(`loanProductId`)  ORDER BY `surname`;  # теперь сгрупируем по названию услуги и выведем количество заявок на каждый вид услуги в 2022 году  SELECT `loanProductName`, COUNT(`loanProductName`) FROM `MainApplication`  INNER JOIN `Person` USING(`personId`)  INNER JOIN `LoanApplication` USING(`loanApplicationId`)  INNER JOIN `LoanProduct` USING(`loanProductId`)  WHERE (`year` = 2022)  GROUP BY `loanProductName`;  # теперь выберем столбец с названием услуги и отсортируем по убыванию количества. После  # при помощи оператора LIMIT органичим количество выводимых строк числом 1. Данный запрос - ответ на второе задание  SELECT `loanProductName` FROM `MainApplication`  INNER JOIN `Person` USING(`personId`)  INNER JOIN `LoanApplication` USING(`loanApplicationId`)  INNER JOIN `LoanProduct` USING(`loanProductId`)  WHERE (`year` = 2022)  GROUP BY `loanProductName`  ORDER BY COUNT(`loanProductName`) DESC  LIMIT 1; |

Непосредственно ответ на задание.

|  |
| --- |
| SELECT `loanProductName` FROM `MainApplication`  INNER JOIN `Person` USING(`personId`)  INNER JOIN `LoanApplication` USING(`loanApplicationId`)  INNER JOIN `LoanProduct` USING(`loanProductId`)  WHERE (`year` = 2022)  GROUP BY `loanProductName`  ORDER BY COUNT(`loanProductName`) DESC  LIMIT 1; |

## Задание 3

Предложите алгоритм для расчета графика аннуитетных платежей, напишите функцию, рассчитывающую график платежей с 5 столбцами: «Месяц», «Ежемесячный платеж», «Основной долг», «Долг по процентам» и «Остаток основного долга».

Ожидаемый результат: словесное описание логики работы функции; функция, которая выводит в строковом формате массив объектов, состоящих из 5 атрибутов указанных выше, на любом языке программирования (JS будет плюсом). Алгоритм должен основываться на модели данных из задания 1.

Дополнительно необходимо указать данные, на основе которых проводилось тестирование работы функции, а также результат её исполнения.

**getAnnuityPaymentsSchedule** рассчитывает график аннуитетных платежей на основании суммы кредита, ставки и срока погашения.

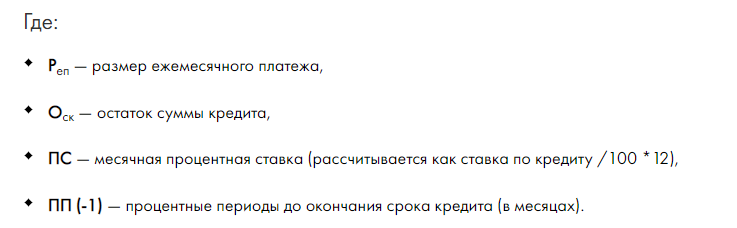
Функция принимает на вход объект, свойствами которого являются сумма кредита, ставка и срок погашения.

Создаются локальные переменные

* debt - остаток по долгу; Изменяется на каждой итерации цикла;
* monthlyInterest - ставка по процентам в месяц; Вычисляется один раз и не изменяется;
* monthlyPayment - ежемесячный платеж; Вычисляется один раз и не изменяется;

Формула расчета аннуитета.





Поэтому, если интерпретировать на требуемую функцию, можно вычислить monthlyPayment следующим образом:

|  |
| --- |
| var monthlyPayment = debt \* ((monthlyInterest)/(1-Math.pow(1+monthlyInterest, -input.loanTerm))); |

( Для возведения в отрицательную степень количества месяцев выражения 1+monthlyInterest была использована функция pow из библиотеки Math).

Далее запускается цикл, в котором совершается обход по месяцам от 1-го до последнего включительно. Долг по процентам рассчитывается как текущий остаток умножить на ежемесячную ставку по процентам.

Основной долг (mainDebt) рассчитывается как разность ежемесячного платежа и долга по процентам.

Остаток основного долга рассчитывается как разность debt и основного долга на каждой итерации.

В функции создаётся массив result, который изначально пуст. На каждой итерации туда вставляется объект со свойствами: Месяц | Eжемесячный платёж | Основной долг | Долг по процентам | Остаток основного долга.

При вставке вещественные числа округляются с точностью 2 знака после запятой. После расчета функция возвращает заполненный массив.

AnnuityPaymentAlgorithm

|  |
| --- |
| let input = {  loanSum: 480000, // сумма кредита  loanInterestRate: 7.5, // ставка  loanTerm: 76 // срок  };  function roundNumber(x){  if((typeof x) === "number") {  if(parseFloat(x.toFixed(2)) === -0)  return 0;  return parseFloat(x.toFixed(2));  }  else  return x;  }  function getAnnuityPaymentsSchedule(input){  let result = [];  var debt = input.loanSum;  var monthlyInterest = (input.loanInterestRate)/(100\*12);  var monthlyPayment = debt \* ((monthlyInterest)/(1-Math.pow(1+monthlyInterest, -input.loanTerm)));  for(let i = 1; i <= input.loanTerm; i++){  let month = i;  let interestDebt = debt\* monthlyInterest;  let mainDebt = monthlyPayment - interestDebt;  debt -= mainDebt;  result.push({  month: month,  monthlyPayment: roundNumber(monthlyPayment),  mainDebt: roundNumber(mainDebt),  interestDebt: roundNumber(interestDebt),  balanceOfPrincipal: roundNumber(debt)  });  }  return result;  }  function printAnnuityPaymentsSchedule(arr){  console.log('Месяц | Eжемесячный платёж | Основной долг | Долг по процентам | Остаток основного долга');  for(let i = 0; i < input.loanTerm; i++) {  console.log(  arr[i].month + ' | ' +  arr[i].monthlyPayment + ' | ' +  arr[i].mainDebt + ' | ' +  arr[i].interestDebt + ' | ' +  arr[i].balanceOfPrincipal  );  }  }  let res = getAnnuityPaymentsSchedule(input);  printAnnuityPaymentsSchedule(res); |

Тестирование

|  |  |
| --- | --- |
| Входные данные | Результат |
| let input = {  loanSum: 100000, // сумма кредита  loanInterestRate: 13, // ставка  loanTerm: 12 // срок  }; | Месяц | Eжемесячный платёж | Основной долг | Долг по процентам | Остаток основного долга  1 | 8931.73 | 7848.39 | 1083.33 | 92151.61  2 | 8931.73 | 7933.42 | 998.31 | 84218.19  3 | 8931.73 | 8019.36 | 912.36 | 76198.82  4 | 8931.73 | 8106.24 | 825.49 | 68092.58  5 | 8931.73 | 8194.06 | 737.67 | 59898.53  6 | 8931.73 | 8282.83 | 648.9 | 51615.7  7 | 8931.73 | 8372.56 | 559.17 | 43243.14  8 | 8931.73 | 8463.26 | 468.47 | 34779.88  9 | 8931.73 | 8554.95 | 376.78 | 26224.94  10 | 8931.73 | 8647.62 | 284.1 | 17577.31  11 | 8931.73 | 8741.31 | 190.42 | 8836  12 | 8931.73 | 8836 | 95.72 | 0 |

|  |  |
| --- | --- |
| Входные данные | Результат |
| let input = {  loanSum: 480000, // сумма кредита  loanInterestRate: 7.5, // ставка  loanTerm: 76 // срок  }; | Месяц | Eжемесячный платёж | Основной долг | Долг по процентам | Остаток основного долга  1 | 7953.45 | 4953.45 | 3000 | 475046.55  2 | 7953.45 | 4984.41 | 2969.04 | 470062.15  3 | 7953.45 | 5015.56 | 2937.89 | 465046.59  4 | 7953.45 | 5046.91 | 2906.54 | 459999.69  5 | 7953.45 | 5078.45 | 2875 | 454921.24  6 | 7953.45 | 5110.19 | 2843.26 | 449811.05  7 | 7953.45 | 5142.13 | 2811.32 | 444668.92  8 | 7953.45 | 5174.27 | 2779.18 | 439494.66  9 | 7953.45 | 5206.6 | 2746.84 | 434288.05  10 | 7953.45 | 5239.15 | 2714.3 | 429048.91  11 | 7953.45 | 5271.89 | 2681.56 | 423777.02  12 | 7953.45 | 5304.84 | 2648.61 | 418472.18  13 | 7953.45 | 5338 | 2615.45 | 413134.18  14 | 7953.45 | 5371.36 | 2582.09 | 407762.82  15 | 7953.45 | 5404.93 | 2548.52 | 402357.89  16 | 7953.45 | 5438.71 | 2514.74 | 396919.18  17 | 7953.45 | 5472.7 | 2480.74 | 391446.48  18 | 7953.45 | 5506.91 | 2446.54 | 385939.58  19 | 7953.45 | 5541.32 | 2412.12 | 380398.25  20 | 7953.45 | 5575.96 | 2377.49 | 374822.3  21 | 7953.45 | 5610.81 | 2342.64 | 369211.49  22 | 7953.45 | 5645.87 | 2307.57 | 363565.62  23 | 7953.45 | 5681.16 | 2272.29 | 357884.45  24 | 7953.45 | 5716.67 | 2236.78 | 352167.79  25 | 7953.45 | 5752.4 | 2201.05 | 346415.39  26 | 7953.45 | 5788.35 | 2165.1 | 340627.04  27 | 7953.45 | 5824.53 | 2128.92 | 334802.51  28 | 7953.45 | 5860.93 | 2092.52 | 328941.58  29 | 7953.45 | 5897.56 | 2055.88 | 323044.02  30 | 7953.45 | 5934.42 | 2019.03 | 317109.6  31 | 7953.45 | 5971.51 | 1981.93 | 311138.09  32 | 7953.45 | 6008.83 | 1944.61 | 305129.25  33 | 7953.45 | 6046.39 | 1907.06 | 299082.87  34 | 7953.45 | 6084.18 | 1869.27 | 292998.69  35 | 7953.45 | 6122.2 | 1831.24 | 286876.48  36 | 7953.45 | 6160.47 | 1792.98 | 280716.01  37 | 7953.45 | 6198.97 | 1754.48 | 274517.04  38 | 7953.45 | 6237.71 | 1715.73 | 268279.33  39 | 7953.45 | 6276.7 | 1676.75 | 262002.63  40 | 7953.45 | 6315.93 | 1637.52 | 255686.7  41 | 7953.45 | 6355.4 | 1598.04 | 249331.29  42 | 7953.45 | 6395.13 | 1558.32 | 242936.17  43 | 7953.45 | 6435.1 | 1518.35 | 236501.07  44 | 7953.45 | 6475.31 | 1478.13 | 230025.76  45 | 7953.45 | 6515.79 | 1437.66 | 223509.97  46 | 7953.45 | 6556.51 | 1396.94 | 216953.46  47 | 7953.45 | 6597.49 | 1355.96 | 210355.98  48 | 7953.45 | 6638.72 | 1314.72 | 203717.26  49 | 7953.45 | 6680.21 | 1273.23 | 197037.04  50 | 7953.45 | 6721.96 | 1231.48 | 190315.08  51 | 7953.45 | 6763.98 | 1189.47 | 183551.1  52 | 7953.45 | 6806.25 | 1147.19 | 176744.85  53 | 7953.45 | 6848.79 | 1104.66 | 169896.06  54 | 7953.45 | 6891.6 | 1061.85 | 163004.46  55 | 7953.45 | 6934.67 | 1018.78 | 156069.79  56 | 7953.45 | 6978.01 | 975.44 | 149091.78  57 | 7953.45 | 7021.62 | 931.82 | 142070.16  58 | 7953.45 | 7065.51 | 887.94 | 135004.65  59 | 7953.45 | 7109.67 | 843.78 | 127894.99  60 | 7953.45 | 7154.1 | 799.34 | 120740.88  61 | 7953.45 | 7198.82 | 754.63 | 113542.07  62 | 7953.45 | 7243.81 | 709.64 | 106298.26  63 | 7953.45 | 7289.08 | 664.36 | 99009.18  64 | 7953.45 | 7334.64 | 618.81 | 91674.54  65 | 7953.45 | 7380.48 | 572.97 | 84294.06  66 | 7953.45 | 7426.61 | 526.84 | 76867.45  67 | 7953.45 | 7473.02 | 480.42 | 69394.43  68 | 7953.45 | 7519.73 | 433.72 | 61874.69  69 | 7953.45 | 7566.73 | 386.72 | 54307.97  70 | 7953.45 | 7614.02 | 339.42 | 46693.94  71 | 7953.45 | 7661.61 | 291.84 | 39032.33  72 | 7953.45 | 7709.49 | 243.95 | 31322.84  73 | 7953.45 | 7757.68 | 195.77 | 23565.16  74 | 7953.45 | 7806.16 | 147.28 | 15759  75 | 7953.45 | 7854.95 | 98.49 | 7904.05  76 | 7953.45 | 7904.05 | 49.4 | 0 |

|  |  |
| --- | --- |
| Входные данные | Результат |
| let input = {  loanSum: 50000, // сумма кредита  loanInterestRate: 9.5, // ставка  loanTerm: 7 // срок  }; | Месяц | Eжемесячный платёж | Основной долг | Долг по процентам | Остаток основного долга  1 | 7370.83 | 6975 | 395.83 | 43025  2 | 7370.83 | 7030.22 | 340.61 | 35994.79  3 | 7370.83 | 7085.87 | 284.96 | 28908.91  4 | 7370.83 | 7141.97 | 228.86 | 21766.94  5 | 7370.83 | 7198.51 | 172.32 | 14568.43  6 | 7370.83 | 7255.5 | 115.33 | 7312.94  7 | 7370.83 | 7312.94 | 57.89 | 0 |