Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный Технический университет»

Кафедра ИИТ

Пояснительная записка

к курсовой работе по дисциплине

«Проектирование баз данных»

по теме

«АРМ Работника рекламного агентства»

КП.ИИ-22.210586 40 03 01

Листов: 35

**Выполнил**

студент 3 курса,

ФЭИС, группы ИИ-22

Копанчук Е.Р.

**Нормоконтроль**

Козик И.Д.

**Проверил**

Козик И.Д.

**Брест 2023**

ВВЕДЕНИЕ …………………………………………………………………………..3

ГЛАВА 1. МОДЕЛЬ ОБЪЕКТА АВТОМАТИЗАЦИИ ...………….………………5

1.1.Информационная модель ……………………………………………………..5

1.2.Функциональная модель ……………………………………………………...8

1.3.Набор текстовых данных для проверки БД ………………………………..10

ГЛАВА 2.РЕЗУЛЬТАТЫ РАЗРАБОТКИ БАЗЫ ДАННЫХ АРМ ……………....13

2.1.Концептуальная модель БД …………………………………………………13

2.2.Логическая модель БД ………………………………………………………13

2.3.Результаты создания, загрузки и проверки БД …………………………….15

ГЛАВА 3. РЕЗУЛЬТАТЫ РАЗРАБОТКИ ПРОГРАММЫ АРМ ………..............28

3.1.Модульная структура и её описание ……………………………………….28

3.2.Описание сертификаций (модулей, запросов, отчетов, форм) …………...28

3.3.Список сообщений …………………………………………………………..29

3.4.Структура информации для Справки ………………………………………30

3.5.СУБД – ориентированная структура программы ……………………….…30

3.6.Структура главного меню АРМ ………………………………………….…30

ГЛАВА 4. РЕЗУЛЬТАТЫ ИСПЫТАНИЙ АРМ …………………………………32

4.1.Цель, задачи и методы испытания ………………………………………….30

4.2.Описание и анализ результатов испытания АРМ …………………………30

ЗАКЛЮЧЕНИЕ: ……………………………………………………………………56

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ………………………………30

СПИСОК СОКРАЩЕНИЙ ………………………………………………………...30

ПРИЛОЖЕНИЯ: ………………………………………………………....................13

А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ АРМ ………………………13

Б. ОПИСАНИЕ ПРИМЕНЕНИЯ АРМ …………………………………………13

В. ИНСТРУКЦИЯ ПО УСТАНОВКЕ И ПРОВЕРКЕ АРМ …………………..15

Г. ТЕКСТ ПРОГРАММЫ ……………………………………………………….15

# 

**ВВЕДЕНИЕ**

В условиях современного рынка массовой информации реклама играет ключевую роль в формировании успешного облика бренда и привлечении внимания целевой аудитории. Рекламные агентства, как организации, специализирующиеся на разработке и реализации рекламных стратегий, становятся неотъемлемой частью бизнес-процессов. В этом контексте, автоматизация и эффективное управление рекламными кампаниями становятся приоритетными задачами, которые успешно решаются с использованием Автоматизированных Рабочих Мест (АРМ) для рекламных агентств.

**Работа Рекламного Агентства:**

Рекламное агентство — это организация, объединяющая специалистов различных профилей для создания и реализации рекламных кампаний. В его структуре ключевые роли принадлежат рекламному директору, медиапланировщику, дизайнеру и клиенту. Рекламный директор отвечает за стратегическое направление рекламных кампаний, медиапланировщик разрабатывает план размещения рекламы, дизайнер создает визуальные материалы, а клиент выступает заказчиком, определяя цели и требования.

**Задачи в Рекламном Бизнесе:**

*Рекламный Директор:* Разработка стратегий, координация команды, анализ рынка и эффективности кампаний.

*Медиапланировщик:* Определение оптимальных каналов и времени размещения рекламы, управление рекламным бюджетом.

*Дизайнер:* Создание эффективных и привлекательных визуальных элементов для рекламных материалов.

*Клиент:* Определение целей, бюджета, контроль качества и эффективности рекламной кампании.

**АРМ в Рекламном Агентстве:**

Автоматизированные Рабочие Места (АРМ) в рекламных агентствах представляют собой системы, обеспечивающие удобное взаимодействие между сотрудниками и обеспечивающие эффективное управление процессами. Курсовой проект посвящен разработке приложения с базой данных, которое позволит создавать, изменять и удалять сущности, а также выполнять поиск по атрибутам. Это приложение облегчит процессы планирования, визуализации и анализа рекламных кампаний, повышая общую производительность и прозрачность внутри агентства.

# ГЛАВА 1. МОДЕЛЬ ОБЪЕКТА АВТОМАТИЗАЦИИ

## 1.1.Информационная модель

**1. Рабочие данные:**

* *Рекламная Компания (Advertising Campaign):*
  + Идентификатор (ID);
  + Название (Name);
  + Дата начала (Start Date);
  + Дата окончания (End Date);
  + Стоимость (Price);
  + Связь с клиентом (ID client);
  + Связь с директором (ID director);
  + Связь с медиапланировщиком (ID media planner);
  + Связь с дизайнером (ID designer).
* *Дизайнер (Designer),*
* *Рекламный Директор (Advertising Director),*
* *Медиапланировщик (Media Planner),*
* *Клиент (Client):*
  + Идентификатор (ID);
  + ФИО (FIO);
  + Email;
  + Телефон (Phone).

**2. Интерфейс пользователя:**

* *Выбор списка сущостей:*
  + Рекламные компании;
  + Клиенты;
  + Рекламные директоры;
  + Медиапланировщики;
  + Дизайнер.
* *Добавление сущности:*
  + Добавить компанию;
  + Добавить клиента;
  + Добавить рекламного директора;
  + Добавить медиапланировщика;
  + Добавить дизайнера.
* *Поиск по базе данных*
  + Найти компанию;
  + Найти персону (Рекламного директора, клиента, медиапланировщика, дизайнера).

## 1.2.Функциональная модель



Рисунок 1.1 – Функциональная модель

При входе в главное меню нашего приложения пользователю представляется возможность выбора из следующих прецедентов программы:

Выбрать список ­­– пользователь может выбрать отображаемы на главном экране список. В каждом списке располагаются экземпляры одной из пяти сущностей (рекламные компании, клиенты, рекламные директоры, медиаланировщики, дизайнеры). После смены списка пользователем, предыдущий список очищается и его наполняют необходимые выбранные сущности.

Открыть элемент – выбрать данную опцию можно нажав на элемент списка, после чего появляется модальное окно демонстрирующее все атрибуты сущности. После просмотра пользователь может закрыть модальное окно попав в главное меню, или же приступить к редактированию сущностей. Для редактирования текстовых полей объектов используются активные edit box, при нажатии на которые можно ввести нужный атрибут. Для изменения полей, которые зависят от существующих данных используются блоки select, при нажатии на которые пользователю предоставляется выбор из возможных вариантов. Такой принцип используется для заполнения полей сущности “Рекламная компания”, а именно её участников можно выбрать прямиком из БД.

Создать элемент – при нажатии на кнопку в нижней части экрана перед нами появляется модальное окно для создания новой сущности, тип создаваемой сущности зависит от типа сущностей выбранного списка. В модальном окне, как и в модальном окне редактирования / просмотра, находятся edit box для ввода текстовых полей и select для заполнения полей с ограниченным вариантом выбора. Создание можно отменить с помощью соответствующей кнопки. Если мы хотим завершить создание, нажимаем кнопку сохранения введенных данных.

Поиск – при нажатии на кнопку поиска появляется модальное окно поиска сущностей. Перед началом пользователю необходимо выбрать, что он собирается найти: рекламную компанию или персону (клиента, директора, медиапланировщика или дизайнера). После чего пользователь может ввести любую строку для поиска, на выход будут поданы сущности, поля которых включают данную строку приведённую в lowercase. После просмотра сущностей, пользователь может закрыть окно поиска или снова ввести запрос.

Завершение – осуществляется при выходе пользователя с сайта приложения.

Запуск – осуществляется при входе пользователя на сайт.  
  
  
**1.3.Набор текстовых данных для проверки БД**

|  |  |
| --- | --- |
| Клиенты:  ФИО: "Клиентова Екатерина Николаева"  Email: "ekaterina.client@gmail.com"  Телефон: "+375440573453"  ФИО: "Клиентов Антон Сергеев"  Email: "anton.client@gmail.com"  Телефон: "+375440573392"  Рекламные директоры:  ФИО: "Боссова Ирина Петрова"  Email: "irina.director@gmail.com"  Телефон: "+375623855530"    ФИО: "Большебоссов Алексей Смирнов"  Email: "alex.director@ gmail.com"  Телефон: "+375837503339" | Медиапланировщики:  ФИО: "Планова Мария Кузнецова"  Email: "maria.planner@ gmail.com"  Телефон: "+375443955562"    ФИО: "План Дмитрий Иванов"  Email: "dmitry.planner@gmail.com"  Телефон: "+375663422234"  Дизайнеры:  ФИО: "Дизайнова Анна Иванова"  Email: "anna.designer@gmail.com"  Телефон: "+375444633303"  ФИО: "Дизайнов Петр Сидоров"  Email: "petr.designer@gmail.com"  Телефон: "+375443204466" |

Рекламные компании:

Название: "Летнее Приключение"

Дата начала: "2023-06-01"

Дата окончания: "2023-08-31"

Стоимость: “2344”

Связь с клиентом: существующий ID клиента

Связь с директором: существующий ID директора

Связь с медиапланировщиком: существующий ID медиапланировщика

Связь с дизайнером: существующий ID дизайнера

Название: "Зимняя Сказка"

Дата начала: "2023-12-01"

Дата окончания: "2024-02-29"

Стоимость: “523”

Связь с клиентом: существующий ID клиента

Связь с директором: существующий ID директора

Связь с медиапланировщиком: существующий ID медиапланировщика

Связь с дизайнером: существующий ID дизайнера

# ГЛАВА 2.РЕЗУЛЬТАТЫ РАЗРАБОТКИ БАЗЫ ДАННЫХ АРМ

## 2.1.Концептуальная модель БД



Рисунок 2.1 – Концептуальная модель БД

**Спецификация сущностей:**

Таблица 2.1 – Спецификация сущностей БД

|  |  |
| --- | --- |
| Компания | Клиент, Рекламный директор, Медиапланировщик, Дизайнер |
| Идентификатор (ID)  Название (Name)  Дата начала (Start Date)  Дата окончания (End Date)  Стоимость (Price)  Связь с клиентом (ID client)  Связь с директором (ID director)  Связь с медиапланировщиком (ID media planner)  Связь с дизайнером (ID designer) | Идентификатор (ID) ФИО (FIO) Email  Телефон (Phone) |

**Спецификация типов связей:**

Клиент (один к многим) Рекламная компания

Рекламный директор (один к многим) Рекламная компания

Медиапланировщик (один к многим) Рекламная компания

Дизайнер (один к многим) Рекламная компания

**Спецификация атрибутов:**

|  |  |
| --- | --- |
| **Компания** | |
| Идентификатор (ID) | SERIAL |
| Название (Name) | VARCHAR(40) |
| Дата начала (Start Date) | DATE |
| Дата окончания (End Date) | DATE |
| Стоимость (Price) | INTEGER |
| Связь с клиентом (ID client) | INTEGER |
| Связь с директором (ID director) | INTEGER |
| Связь с медиапланировщиком (ID media planner) | INTEGER |
| Связь с дизайнером (ID designer) | INTEGER |
| **Клиент, Рекламный директор, Медиапланировщик, Дизайнер** | |
| Идентификатор (ID) | SETIAL |
| ФИО (FIO) | VARCHAR(40) |
| Email | VARCHAR(60) |
| Телефон (Phone) | VARCHAR(15) |

Таблица 2.2 – Спецификация атрибутов

## 2.2.Логическая модель БД



Рисунок 2.2 – Логическая модель БД

Логическая модель базы данных (Рисунок 2.2) для рекламного агентства представляет собой структуру, состоящую из пяти взаимосвязанных таблиц. Каждая таблица включает уникальный идентификатор (ID), который автоматически генерируется и является первичным ключом. Первые четыре таблицы предназначены для хранения информации о дизайнерах, рекламных директорах, медиапланировщиках и клиентах, включая их ФИО, электронные адреса и телефоны. Внешние ключи (ID\_Client, ID\_Director, ID\_Media\_Planner, ID\_Designer) в таблице Advertising\_Campaign связывают рекламные компании с соответствующими участниками. Таблица Advertising\_Campaign содержит информацию о рекламных компаниях, такую как название, даты начала и окончания, а также связи с клиентом, директором, медиапланировщиком и дизайнером. Связи в этой модели позволяют эффективно организовать информацию и обеспечивают целостность данных. Типы данных, такие как VARCHAR для строковых значений и DATE для дат, выбраны с учетом характера хранимой информации. Модель обеспечивает гибкость и расширяемость для удовлетворения потребностей бизнеса в будущем.

## 2.3. Результаты создания, загрузки и проверки БД

mydb=# SELECT \* FROM client;

id | fio | email | phone

------+-----------------------------------------------+--------------------------------------+-------------------------

5 | Клиентова Екатерина Николаева | ekaterina.client@gmail.com | +375440573453

6 | Клиентов Антон Сергеев | anton.client@gmail.com | +375440573392

(2 строки)

mydb=# SELECT \* FROM director;

id | fio | email | phone

------+-----------------------------------------------+--------------------------------------+-------------------------

12 | Боссова Ирина Петрова | irina.director@gmail.com | +375623855530

13 | Большебоссов Алексей Смирнов | alex.director@gmail.com | +375837503339

(2 строки)

mydb=# SELECT \* FROM planner;

id | fio | email | phone

------+-----------------------------------------------+--------------------------------------+-------------------------

4 | Планова Мария Кузнецова | maria.planner@ gmail.com | +375443955562

5 | План Дмитрий Иванов | dmitry.planner@gmail.com | +375663422234

(2 строки)

mydb=# SELECT \* FROM designer;

id | fio | email | phone

------+-----------------------------------------------+--------------------------------------+-------------------------

3 | Дизайнова Анна Иванова | anna.designer@gmail.com | +375444633303

4 | Дизайнов Петр Сидоров | petr.designer@gmail.com | +375443204466 (2 строки)

(2 строки)

mydb=# SELECT \* FROM company;

id | name | srt\_date | end\_date | price | id\_client | id\_director | id\_planner | id\_designer

----+-------------------------------+-----------------+----------------+--------+-------------+----------------+---------------+-----------------

8 | Летнее Приключение | 2023-06-01 | 2023-08-31 | 2344 | 6 | 12 | 5 | 3

9 | Зимняя Сказка | 2023-12-01 | 2024-02-29 | 523 | 5 | 13 | 4 | 4

(2 строки)

# ГЛАВА 3.РЕЗУЛЬТАТЫ РАЗРАБОТКИ ПРОГРАММЫ АРМ

## 3.1.Модульная структура и её описание

В ходе разработки программы автоматизированной рабочей среды (АРМ) для рекламного агентства была реализована модульная структура, обеспечивающая гибкость, масштабируемость и эффективное управление различными аспектами бизнес-процессов.

**Модуль выбора списка просмотра**

Этот модуль предоставляет пользователю удобный механизм выбора сущности для просмотра или редактирования. Использование элемента select обеспечивает легкость навигации и быстрый доступ к различным видам данных. Пользователь может мгновенно переключаться между списками, что улучшает общую эффективность использования приложения.

**Модуль создания сущности**

Использование модального окна для создания сущности обеспечивает четкость и простоту процесса. Пользователь может в удобной форме ввести необходимые данные, а использование элементов edit box и select делает процесс ввода интуитивно понятным. Модальное окно позволяет сосредоточить внимание пользователя на текущей задаче, минимизируя внешние воздействия.

**Модуль просмотра сущности**

Предоставление дополнительной информации о сущности по запросу пользователя через модальное окно обеспечивает глубокое понимание данных. Пользователь может просматривать подробную информацию о выбранной сущности, улучшая тем самым общую наглядность и удобство использования приложения.

**Модуль поиска сущности**

Модуль поиска позволяет пользователям быстро и эффективно находить нужные данные. Использование модального окна с возможностью выбора типа сущности и ввода критериев поиска делает процесс поиска максимально гибким. Результаты поиска могут быть быстро просмотрены, что экономит время пользователя и упрощает процесс поиска нужных данных.

**Модуль редактирования сущностей**

Модуль редактирования сущностей обеспечивает возможность быстро и удобно вносить изменения в данные. Использование активных edit box и select при редактировании текстовых полей и полей с ограниченным выбором упрощает процесс внесения изменений. Пользователь может мгновенно просматривать и сохранять изменения, что способствует оперативному управлению данными.

В целом, эти модули значительно улучшают пользовательский опыт, обеспечивая простоту взаимодействия с приложением, повышая его функциональность и ускоряя выполнение повседневных задач.

## 3.2.Описание сертификаций (модулей, запросов, отчетов, форм)

**getAllCompanies: Получение всех компаний:** Эта функция извлекает все компании из базы данных и возвращает их в формате JSON. Предназначена для использования при необходимости отобразить список всех зарегистрированных компаний.

**getCompany: Получение компании по идентификатору:** Данная функция принимает идентификатор компании в качестве параметра, выполняет запрос в базу данных и возвращает информацию о компании с указанным идентификатором в формате JSON. Используется для отображения подробной информации о конкретной компании.

**createCompany: Создание новой компании:** Эта функция принимает данные о новой компании, вставляет их в базу данных и возвращает информацию о только что созданной компании в формате JSON. Используется при регистрации новых компаний.

**updateCompany: Обновление данных компании:** Данная функция обновляет информацию о компании на основе предоставленных данных. Принимает идентификатор компании и новые данные, выполняет обновление в базе данных и возвращает обновленную информацию о компании в формате JSON.

**deleteCompany: Удаление компании:** Эта функция удаляет компанию с указанным идентификатором из базы данных. При удалении также обнуляются связанные с этой компанией данные в других таблицах. Возвращает информацию о удаленной компании в формате JSON, а также связанные с ней данные.

**getAllClients: Получение всех клиентов:** Эта функция извлекает все данные о клиентах из базы данных и возвращает их в формате JSON. Предназначена для использования при отображении списка всех зарегистрированных клиентов.

**getClient: Получение клиента по идентификатору:** Функция принимает идентификатор клиента в качестве параметра, выполняет запрос в базу данных и возвращает информацию о клиенте с указанным идентификатором в формате JSON. Используется для отображения подробной информации о конкретном клиенте.

**createClient: Создание нового клиента:** Эта функция принимает данные о новом клиенте, вставляет их в базу данных и возвращает информацию о только что созданном клиенте в формате JSON. Используется при регистрации новых клиентов.

**updateClient: Обновление данных клиента:** Функция обновляет информацию о клиенте на основе предоставленных данных. Принимает идентификатор клиента и новые данные, выполняет обновление в базе данных и возвращает обновленную информацию о клиенте в формате JSON.

**deleteClient: Удаление клиента:** Эта функция удаляет клиента с указанным идентификатором из базы данных. При удалении также обнуляются связанные с этим клиентом данные в других таблицах. Возвращает информацию о удаленном клиенте в формате JSON, а также связанные с ним данные.

**getAllDirectors: Получение всех директоров:** Функция извлекает все данные о директорах из базы данных и возвращает их в формате JSON. Используется для отображения списка всех зарегистрированных директоров.

**getDirector: Получение директора по идентификатору:** Данная функция принимает идентификатор директора в качестве параметра, выполняет запрос в базу данных и возвращает информацию о директоре с указанным идентификатором в формате JSON. Используется для отображения подробной информации о конкретном директоре.

**createDirector: Создание нового директора:** Эта функция принимает данные о новом директоре, вставляет их в базу данных и возвращает информацию о только что созданном директоре в формате JSON. Используется при регистрации новых директоров.

**updateDirector: Обновление данных директора:** Функция обновляет информацию о директоре на основе предоставленных данных. Принимает идентификатор директора и новые данные, выполняет обновление в базе данных и возвращает обновленную информацию о директоре в формате JSON.

**deleteDirector: Удаление директора:** Эта функция удаляет директора с указанным идентификатором из базы данных. При удалении также обнуляются связанные с этим директором данные в других таблицах. Возвращает информацию о удаленном директоре в формате JSON, а также связанные с ним данные.

**getAllPlanners: Получение всех медиапланировщиков:** Данная функция извлекает все данные о медиапланировщиках из базы данных и возвращает их в формате JSON. Используется для отображения списка всех зарегистрированных медиапланировщиков.

**getPlanner: Получение медиапланировщика по идентификатору:** Функция принимает идентификатор медиапланировщика в качестве параметра, выполняет запрос в базу данных и возвращает информацию о медиапланировщике с указанным идентификатором в формате JSON. Используется для отображения подробной информации о конкретном медиапланировщике.

**createPlanner: Создание нового медиапланировщика:** Эта функция принимает данные о новом медиапланировщике, вставляет их в базу данных и возвращает информацию о только что созданном медиапланировщике в формате JSON. Используется при регистрации новых медиапланировщиков.

**updatePlanner: Обновление данных медиапланировщика:** Функция обновляет информацию о медиапланировщике на основе предоставленных данных. Принимает идентификатор медиапланировщика и новые данные, выполняет обновление в базе данных и возвращает обновленную информацию о медиапланировщике в формате JSON.

**deletePlanner: Удаление медиапланировщика:** Эта функция удаляет медиапланировщика с указанным идентификатором из базы данных. При удалении также обнуляются связанные с этим медиапланировщиком данные в других таблицах. Возвращает информацию о удаленном медиапланировщике в формате JSON, а также связанные с ним данные.

**getAllDesigners: Получение всех дизайнеров:** Функция извлекает все данные о дизайнерах из базы данных и возвращает их в формате JSON. Используется для отображения списка всех зарегистрированных дизайнеров.

**getDesigner: Получение дизайнера по идентификатору:** Данная функция принимает идентификатор дизайнера в качестве параметра, выполняет запрос в базу данных и возвращает информацию о дизайнере с указанным идентификатором в формате JSON. Используется для отображения подробной информации о конкретном дизайнере.

**createDesigner: Создание нового дизайнера:** Эта функция принимает данные о новом дизайнере, вставляет их в базу данных и возвращает информацию о только что созданном дизайнере в формате JSON. Используется при регистрации новых дизайнеров.

**updateDesigner: Обновление данных дизайнера:** Функция обновляет информацию о дизайнере на основе предоставленных данных. Принимает идентификатор дизайнера и новые данные, выполняет обновление в базе данных и возвращает обновленную информацию о дизайнере в формате JSON.

**deleteDesigner: Удаление дизайнера:** Эта функция удаляет дизайнера с указанным идентификатором из базы данных. При удалении также обнуляются связанные с этим дизайнером данные в других таблицах. Возвращает информацию о удаленном дизайне

**searchCompany: Поиск компаний:** Функция выполняет поиск компаний по заданной подстроке в различных полях сущности "company". Принимает строку поиска в качестве параметра и возвращает результаты поиска в формате JSON. Используется для поиска компаний по различным критериям, таким как название, даты, цена и идентификаторы связанных сущностей.

**searchClient: Поиск клиентов:** Данная функция осуществляет поиск клиентов по указанной подстроке в полях "fio", "email" и "phone" сущности "client". Принимает строку поиска в качестве параметра и возвращает результаты поиска в формате JSON. Используется для поиска клиентов по ФИО, электронной почте и телефону.

**searchDirector: Поиск директоров:** Эта функция выполняет поиск директоров по указанной подстроке в полях "fio", "email" и "phone" сущности "director". Принимает строку поиска в качестве параметра и возвращает результаты поиска в формате JSON. Используется для поиска директоров по ФИО, электронной почте и телефону.

**searchPlanner: Поиск медиапланировщиков:** Функция осуществляет поиск медиапланировщиков по указанной подстроке в полях "fio", "email" и "phone" сущности "planner". Принимает строку поиска в качестве параметра и возвращает результаты поиска в формате JSON. Используется для поиска медиапланировщиков по ФИО, электронной почте и телефону.

**searchDesigner: Поиск дизайнеров:** Данная функция выполняет поиск дизайнеров по заданной подстроке в полях "fio", "email" и "phone" сущности "designer". Принимает строку поиска в качестве параметра и возвращает результаты поиска в формате JSON. Используется для поиска дизайнеров по ФИО, электронной почте и телефону.

## 3.3.Список сообщений

| **Запрос** | **SQL** |
| --- | --- |
| Выбор всех компаний | SELECT \* FROM company; |
| Выбор всех клиентов | SELECT \* FROM client; |
| Выбор всех директоров | SELECT \* FROM director; |
| Выбор всех медиапланировщиков | SELECT \* FROM planner; |
| Выбор всех дизайнеров | SELECT \* FROM designer; |
| Поиск среди полей сущности "company" | SELECT \* FROM company WHERE "name" ILIKE $1 OR srt\_date::TEXT ILIKE $1 OR end\_date::TEXT ILIKE $1 OR price::TEXT ILIKE $1 OR id\_client::TEXT ILIKE $1 OR id\_director::TEXT ILIKE $1 OR id\_planner::TEXT ILIKE $1 OR id\_designer::TEXT ILIKE $1; |
| Поиск среди полей сущности "client" | SELECT \* FROM client WHERE fio ILIKE $1 OR email ILIKE $1 OR phone ILIKE $1; |
| Поиск среди полей сущности "director" | SELECT \* FROM director WHERE fio ILIKE $1 OR email ILIKE $1 OR phone ILIKE $1; |
| Поиск среди полей сущности "planner" | SELECT \* FROM planner WHERE fio ILIKE $1 OR email ILIKE $1 OR phone ILIKE $1; |
| Поиск среди полей сущности "designer" | SELECT \* FROM designer WHERE fio ILIKE $1 OR email ILIKE $1 OR phone ILIKE $1; |
| Выбор компании по идентификатору | SELECT \* FROM company WHERE id = $1; |
| Выбор клиента по идентификатору | SELECT \* FROM client WHERE id = $1; |
| Выбор директора по идентификатору | SELECT \* FROM director WHERE id = $1; |
| Выбор медиапланировщика по идентификатору | SELECT \* FROM planner WHERE id = $1; |
| Выбор дизайнера по идентификатору | SELECT \* FROM designer WHERE id = $1; |
| Вставка новой компании | INSERT INTO company (name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer) VALUES ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*; |
| Вставка нового клиента | INSERT INTO client (fio, email, phone) VALUES ($1, $2, $3) RETURNING \*; |
| Вставка нового директора | INSERT INTO director (fio, email, phone) VALUES ($1, $2, $3) RETURNING \*; |
| Вставка нового медиапланировщика | INSERT INTO planner (fio, email, phone) VALUES ($1, $2, $3) RETURNING \*; |
| Вставка нового дизайнера | INSERT INTO designer (fio, email, phone) VALUES ($1, $2, $3) RETURNING \*; |
| Обновление данных компании | UPDATE company SET name = $1, srt\_date = $2, end\_date = $3, price = $4, id\_client = $5, id\_director = $6, id\_planner = $7, id\_designer = $8 WHERE id = $9 RETURNING \*; |
| Обновление данных клиента | UPDATE client SET fio = $1, email = $2, phone = $3 WHERE id = $4 RETURNING \*; |
| Обновление данных директора | UPDATE director SET fio = $1, email = $2, phone = $3 WHERE id = $4 RETURNING \*; |
| Обновление данных медиапланировщика | UPDATE planner SET fio = $1, email = $2, phone = $3 WHERE id = $4 RETURNING \*; |
| Обновление данных дизайнера | UPDATE designer SET fio = $1, email = $2, phone = $3 WHERE id = $4 RETURNING \*; |
| Удаление компании по идентификатору | DELETE FROM company WHERE id = $1; |
| Удаление клиента по идентификатору | DELETE FROM client WHERE id = $1; |
| Удаление директора по идентификатору | DELETE FROM director WHERE id = $1; |
| Удаление медиапланировщика по идентификатору | DELETE FROM planner WHERE id = $1; |
| Удаление дизайнера по идентификатору | DELETE FROM designer WHERE id = $1; |
|  |  |

## 3.4.Структура информации для справки

**Клиент**

{

"id": 5,

"fio": "Клиентова Екатерина Николаева",

"email": "ekaterina.client@gmail.com",

"phone": "+375440573453"

}

| **Название** | **Тип** | **Описание** |
| --- | --- | --- |
| id | SERIAL | Уникальный идентификатор клиента. Автоматически увеличиваемое целочисленное значение, используемое для идентификации конкретного клиента в базе данных. |
| fio | VARCHAR(40) | ФИО клиента. Максимальная длина 40 символов. |
| email | VARCHAR(60) | Адрес электронной почты клиента. Максимальная длина 60 символов. |
| phone | VARCHAR(15) | Телефонный номер клиента. Максимальная длина 15 символов. |

**Рекламный директор**

{

"id": 12,

"fio": "Боссова Ирина Петрова",

"email": "irina.director@gmail.com",

"phone": "+375623855530"

}

| **Название** | **Тип** | **Описание** |
| --- | --- | --- |
| id | SERIAL | Уникальный идентификатор директора. Автоматически увеличиваемое целочисленное значение, используемое для идентификации конкретного директора в базе данных. |
| fio | VARCHAR(40) | ФИО директора. Максимальная длина 40 символов. |
| email | VARCHAR(60) | Адрес электронной почты директора. Максимальная длина 60 символов. |
| phone | VARCHAR(15) | Телефонный номер директора. Максимальная длина 15 символов. |

**Медиапланировщик**

{

"id": 4,

"fio": "Планова Мария Кузнецова",

"email": "maria.planner@gmail.com",

"phone": "+375443955562"

}

| **Название** | **ТИП** | **Описание** |
| --- | --- | --- |
| id | SERIAL | Уникальный идентификатор медиапланировщика. Автоматически увеличиваемое целочисленное значение, используемое для идентификации конкретного медиапланировщика в базе данных. |
| fio | VARCHAR(40) | ФИО медиапланировщика. Максимальная длина 40 символов. |
| email | VARCHAR(60) | Адрес электронной почты медиапланировщика. Максимальная длина 60 символов. |
| phone | VARCHAR(15) | Телефонный номер медиапланировщика. Максимальная длина 15 символов. |

**Дизайнер**

{

"id": 3,

"fio": "Дизайнова Анна Иванова",

"email": "anna.designer@gmail.com",

"phone": "+375444633303"

}

| **Название** | **ТИП** | **Описание** |
| --- | --- | --- |
| id | SERIAL | Уникальный идентификатор дизайнера. Автоматически увеличиваемое целочисленное значение, используемое для идентификации конкретного дизайнера в базе данных. |
| fio | VARCHAR(40) | ФИО дизайнера. Максимальная длина 40 символов. |
| email | VARCHAR(60) | Адрес электронной почты дизайнера. Максимальная длина 60 символов. |
| phone | VARCHAR(15) | Телефонный номер дизайнера. Максимальная длина 15 символов. |

**Рекламная компания**

{

"id": 8,

"name": "Летнее Приключение",

"srt\_date": "2023-06-01",

"end\_date": "2023-08-31",

"price": 2344,

"id\_client": 6,

"id\_director": 12,

"id\_planner": 5,

"id\_designer": 3

}

| **Название** | **ТИП** | **Описание** |
| --- | --- | --- |
| id | SERIAL | Уникальный идентификатор компании. Автоматически увеличиваемое целочисленное значение, используемое для идентификации конкретной компании в базе данных. |
| name | VARCHAR(40) | Название компании. Максимальная длина 40 символов. |
| srt\_date | DATE | Дата начала деятельности компании. |
| end\_date | DATE | Дата окончания деятельности компании. |
| price | INTEGER | Цена услуг компании. |
| id\_client | INTEGER | Идентификатор связанного клиента. |
| id\_director | INTEGER | Идентификатор связанного директора. |
| id\_planner | INTEGER | Идентификатор связанного медиапланировщика. |
| id\_designer | INTEGER | Идентификатор связанного дизайнера. |

## 3.5.СУБД – ориентированная структура программы

Система управления базами данных (СУБД) PostgreSQL была выбрана в качестве основной технологии для реализации базы данных в рамках разработки программы АРМ для рекламного агентства. Этот выбор обусловлен несколькими преимуществами PostgreSQL. Во-первых, PostgreSQL предоставляет расширенные возможности для обработки сложных структур данных и поддерживает широкий спектр типов данных. Во-вторых, СУБД обеспечивает высокий уровень надежности и целостности данных, что критически важно для приложения, взаимодействующего с конфиденциальной информацией клиентов.

Клиент-серверная архитектура разработанного сайта представляет собой взаимодействие четырех ключевых компонентов. База данных (БД) предоставляет хранение и организацию данных, используемых в приложении. PostgreSQL, как СУБД, обеспечивает эффективное взаимодействие с данными, обеспечивает транзакционность и защиту информации.

Сервер, реализованный на Node.js с использованием фреймворка Express, служит промежуточным звеном между клиентской частью сайта и базой данных. Этот компонент обрабатывает запросы от клиента, взаимодействует с СУБД, и предоставляет клиенту необходимые данные через API. Это обеспечивает эффективное управление ресурсами и высокую производительность системы.

Клиентская часть, реализованная с использованием Next.js, представляет собой интерфейс для конечного пользователя. Этот компонент обеспечивает визуализацию данных, взаимодействие с пользователем и отправку запросов на сервер для получения необходимой информации.

Такая архитектура позволяет эффективно управлять данными рекламного агентства, обеспечивает быстрый доступ к информации и поддерживает надежность и безопасность операций с использованием PostgreSQL в качестве СУБД.

## 3.6.Структура главного меню АРМ

# 

**Модуль выбора списка просмотра (Рисунок 6.1):** Этот модуль предоставляет удобный механизм выбора сущности для просмотра или редактирования. Использование элемента select обеспечивает легкость навигации и быстрый доступ к различным видам данных. Пользователь может мгновенно переключаться между списками, что улучшает общую эффективность использования приложения. Интерфейс модуля включает элемент select с перечислением доступных списков.

**Модуль создания сущности (Рисунок 6.2):** Использование модального окна для создания сущности обеспечивает четкость и простоту процесса. Пользователь может в удобной форме ввести необходимые данные, а использование элементов edit box и select делает процесс ввода интуитивно понятным. Модальное окно позволяет сосредоточить внимание пользователя на текущей задаче, минимизируя внешние воздействия. Интерфейс модуля включает модальное окно с edit box и select для ввода данных.

**Модуль просмотра сущности (Рисунок 6.3):** Предоставление дополнительной информации о сущности по запросу пользователя через модальное окно обеспечивает глубокое понимание данных. Пользователь может просматривать подробную информацию о выбранной сущности, улучшая тем самым общую наглядность и удобство использования приложения. Интерфейс модуля включает модальное окно с подробной информацией о сущности.

**Модуль поиска сущности (Рисунок 6.4):** Модуль поиска позволяет пользователям быстро и эффективно находить нужные данные. Использование модального окна с возможностью выбора типа сущности и ввода критериев поиска делает процесс поиска максимально гибким. Результаты поиска могут быть быстро просмотрены, что экономит время пользователя и упрощает процесс поиска нужных данных. Интерфейс модуля включает модальное окно с выбором типа сущности, вводом критериев и результатами поиска.

# ГЛАВА 4.РЕЗУЛЬТАТЫ ИСПЫТАНИЯ АРМ

## 4.1.Цель, задачи и методы испытаний

Целью испытаний Автоматизированной Рабочей Среды (АРС) является проверка ее функциональности, надежности и эффективности, а также оценка соответствия разработанной системы заявленным требованиям. Испытания направлены на выявление потенциальных проблем, уточнение характеристик производительности и обеспечение стабильной работы приложения в различных сценариях использования. Основными задачами испытаний являются проверка корректности работы функциональных модулей, анализ отклика системы на нагрузку, тестирование восстановления после сбоев, и обеспечение безопасности данных пользователей.

Методы испытаний включают в себя функциональное тестирование, где проверяется соответствие функций системы заявленным требованиям. Проверяется удобство использования и выявления потенциальных проблем с точки зрения конечного пользователя.

Испытания проводятся в условиях, максимально приближенных к реальным сценариям использования. Это включает в себя тестирование в различных окружениях, на разных устройствах и под разными операционными системами. Используются сценарии, созданные с учетом возможных вариантов взаимодействия пользователя с системой, а также ситуаций, которые могут привести к потенциальным сбоям.

Результаты испытаний позволят выявить и устранить выявленные проблемы, гарантируя стабильную и эффективную работу Автоматизированной Рабочей Среды. Также они будут использованы для оптимизации производительности и улучшения общего пользовательского опыта при работе с приложением.

## 4.2.Описание и оценка результатов испытания АРМ

Проведение испытаний АРМ позволило получить всестороннюю картину функционирования системы в различных сценариях использования. В ходе функционального тестирования была проверена корректность работы каждого функционального модуля, исключены выявленные ошибки и гарантирована стабильность работы системы.

Пользовательское тестирование подтвердило удобство интерфейса и интуитивную понятность работы с системой. Обратная связь пользователей была положительной, и выявленные моменты были внесены в список улучшений для будущих релизов.

Общий результат испытаний Автоматизированной Рабочей Среды оценивается как успешный. Система демонстрирует высокую степень надежности, производительности и безопасности, что делает ее эффективным инструментом для работы рекламного агентства. Полученные результаты послужат основой для дальнейшего совершенствования и развития АРМ, а также оптимизации ее использования в повседневной деятельности агентства.

|  |  |
| --- | --- |
|  |  |
| Рисунок 2.1 - Просмотр сущности | Рисунок 2.2 - Поиск |
|  |  |
|  |  |
| Рисунок 2.3 - Выбор другого списка | Рисунок 2.4 - Редактирование сущности |

# ЗАКЛЮЧЕНИЕ:

В ходе разработки Автоматизированной Рабочей Среды (АРМ) для рекламного агентства были реализованы функциональные возможности, направленные на повышение эффективности управления данными и процессами в агентстве. Система предоставляет пользователям удобный интерфейс для работы с различными сущностями, такими как рекламные компании, клиенты, рекламные директоры, медиапланировщики и дизайнеры. Выбор PostgreSQL в качестве системы управления базами данных обеспечил надежное хранение и организацию данных. PostgreSQL предоставляет богатые возможности для обработки сложных структур данных и поддерживает разнообразные типы данных, что критически важно для работы с различными аспектами рекламного бизнеса. Высокий уровень надежности и целостности данных PostgreSQL способствует обеспечению безопасности конфиденциальной информации клиентов. Клиент-серверная архитектура, основанная на взаимодействии четырех ключевых компонентов (базы данных, сервера, клиента и СУБД PostgreSQL), обеспечивает эффективное управление ресурсами и высокую производительность системы. Node.js с использованием Express фреймворка успешно служит промежуточным звеном между клиентской частью сайта и базой данных, обрабатывая запросы и предоставляя необходимую информацию через API.

В ходе выполнения данного курсового проекта, я приобрел ценный опыт работы с системой управления базами данных (СУБД) реляционного типа, в моем случае – PostgreSQL. Этот опыт оказался особенно полезным, поскольку я научился эффективно организовывать и хранить данные, а также взаимодействовать с ними в рамках приложений.

Одним из ключевых навыков, которым я освоился, стал процесс проектирования и создания структуры базы данных. PostgreSQL предоставил мне возможность работать с разнообразными типами данных и строить сложные связи между таблицами, что оказалось крайне важным в контексте управления информацией для рекламного агентства.

Внедрение базы данных в приложение также стало неотъемлемой частью моего обучения. Я научился создавать запросы для извлечения данных, модификации и добавления новой информации. Этот опыт был ключевым для обеспечения эффективного взаимодействия между серверной и клиентской частями приложения.

Кроме того, работа с PostgreSQL научила меня обеспечивать целостность данных и безопасность информации. Я усвоил принципы транзакционности и обработки ошибок, что является фундаментальным аспектом при построении надежных и устойчивых приложений.

В целом, опыт работы с PostgreSQL в ходе этого проекта не только расширил мой технический арсенал, но и предоставил практические знания, которые можно применить в будущих проектах. Этот курсовой проект стал важным этапом в моем профессиональном развитии, позволив освоить ключевые аспекты работы с базами данных в реальных условиях разработки приложений.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ:

1. <https://expressjs.com/> (дата доступа 25.11.23 19:00)
2. <https://www.postgresql.org/> (дата доступа 25.11.23 19:00)
3. <https://nextjs.org/> (дата доступа 25.11.23 19:00)

**СПИСОК СОКРАЩЕНИЙ**

**Система обработки данных СОД** — набор аппаратных и программных средств осуществляющих выполнение задач по управлению данными.

**Предметная область ПрО** — часть реального мира подлежащая изучению с целью организации управления и в конечном итоге автоматизации.

**База данных БД** — совокупность данных по определенным правилам предусматривающая общие принципы описания данных, хранения, манипулирование данными и БД является независимой от прикладных программ.

**Система управления БД СУБД** —совокупность программ и языковых средств, предназначенных для управления данными в БД и для обеспечения взаимодействия ее с прикладными программами.

**Банк данных БнД** — автоматизированная информационная система, включающая свойства БД и СУБД.

Разработанная Автоматизированная Рабочая Среда (АРМ) для рекламного  
агентства предоставляет эффективные средства управления данными и автоматизации бизнес-процессов. При входе в главное меню приложения пользователь сталкивается с рядом функций, направленных на оптимизацию работы с информацией о рекламных компаниях, клиентах, рекламных директорах, медиапланировщиках и дизайнерах.

Функционал "Выбрать список" обеспечивает возможность быстрого переключения между списками, предоставляя пользователю доступ к экземплярам сущностей выбранного типа. Эта функция осуществляется на основе хранения данных в базе данных PostgreSQL и предоставляет возможность динамичного отображения актуальной информации.

"Открыть элемент" позволяет пользователям подробно изучать атрибуты выбранной сущности. Взаимодействие с данными осуществляется через активные edit box и блоки select, обеспечивая гибкость в редактировании и обновлении информации. Это обеспечивает точное и быстрое внесение изменений в данные.

Функция "Создать элемент" предоставляет возможность добавления новых сущностей с использованием модального окна. Edit box и select обеспечивают простой и интуитивно понятный способ ввода данных, которые затем сохраняются в базе данных. Этот процесс предоставляет гарантированную целостность и сохранность данных.

"Поиск" представляет собой мощный инструмент для быстрого обнаружения нужной информации. Пользователь может выбрать тип сущности, ввести строку для поиска, и система отобразит все соответствующие результаты. Это обеспечивает высокую эффективность поиска и позволяет быстро находить необходимые данные.

1. Загрузить проект из открытого github репозитория с помощью команды [*https://github.com/Corowka/armadv.git*](https://github.com/Corowka/armadv.git) *(дата доступа 25.11.23 19:00).*

2. Подгрузить внешние зависимости и запустить сервер приложения войдя в папку */server* с помощью команд npm install; *npm start*  
3. Подгрузить внешние зависимости и запустить клиет приложения войдя в папку */client* с помощью команды npm install; npm run dev

4. Зайти на страницу браузера с адресом *localhost:8080*

const db = require('../db')

class Controller {

// Company

async getAllCompanies(req, res) {

const companies = await db.query('SELECT \* FROM company')

res.json(companies.rows)

}

async getCompany(req, res) {

const id = rea.params.id

const company = await db.query('SELECT \* FROM company where id = $1', [id])

res.json(user.rows[0])

}

async createCompany(req, res) {

const { name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer } = req.body

const newCompany = await db.query(

`INSERT INTO company

(name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer)

values ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*`,

[name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer]

)

res.json(newCompany.rows[0])

}

async updateCompany(req, res) {

const { id, name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer } = req.body;

const company = await db.query(

`UPDATE company set

name = $1, srt\_date = $2, end\_date = $3, price = $4,

id\_client = $5, id\_director = $6, id\_planner = $7, id\_designer = $8

where id = $9 RETURNING \*`,

[name, srt\_date, end\_date, price, id\_client, id\_director, id\_planner, id\_designer, id]

)

res.json(company.rows[0])

}

async deleteCompany(req, res) {

const id = req.params.id

const company = await db.query('DELETE FROM company where id = $1', [id])

res.json(company.rows[0])

}

// Client

async getAllClients(req, res) {

const clients = await db.query('SELECT \* FROM client')

res.json(clients.rows)

}

async getClient(req, res) {

body

const id = rea.params.id

const client = await db.query('SELECT \* FROM client where id = $1', [id])

res.json(client.rows[0])

}

async createClient(req, res) {

const { fio, email, phone } = req.body

const newClient = await db.query(

`INSERT INTO client

(fio, email, phone)

values ($1, $2, $3) RETURNING \*`,

[fio, email, phone]

)

res.json(newClient.rows[0])

}

async updateClient(req, res) {

const { id, fio, email, phone } = req.body

const client = await db.query(

`UPDATE client set

fio = $1, email = $2, phone = $3

where id = $4 RETURNING \*`,

[fio, email, phone, id]

)

res.json(client.rows[0])

}

async deleteClient(req, res) {

const id = req.params.id

const companies = await db.query(

`UPDATE company

SET id\_client = NULL

WHERE id\_client = $1 RETURNING \*`,

[id]

)

const client = await db.query('DELETE FROM director where id = $1', [id])

res.json({client: client.rows[0], companies: companies.rows})

}

// Director

async getAllDirectors(req, res) {

const directors = await db.query('SELECT \* FROM director')

res.json(directors.rows)

}

async getDirector(req, res) {

const id = rea.params.id

const director = await db.query('SELECT \* FROM client where id = $1', [id])

res.json(director.rows[0])

}

async createDirector(req, res) {

const { fio, email, phone } = req.body

const newDirector = await db.query(

`INSERT INTO director

(fio, email, phone)

values ($1, $2, $3) RETURNING \*`,

[fio, email, phone]

)

res.json(newDirector.rows[0])

}

async updateDirector(req, res) {

const { id, fio, email, phone } = req.body

const director = await db.query(

`UPDATE director set

fio = $1, email = $2, phone = $3

where id = $4 RETURNING \*`,

[fio, email, phone, id]

)

res.json(director.rows[0])

}

async deleteDirector(req, res) {

const id = req.params.id

const companies = await db.query(

`UPDATE company

SET id\_director = NULL

WHERE id\_director = $1 RETURNING \*`,

[id]

)

const director = await db.query('DELETE FROM director where id = $1', [id])

res.json({director: director.rows[0], companies: companies.rows})

}

// Planner

async getAllPlanners(req, res) {

const planners = await db.query('SELECT \* FROM planner')

res.json(planners.rows)

}

async getPlanner(req, res) {

const id = rea.params.id

const planner = await db.query('SELECT \* FROM planner where id = $1', [id])

res.json(planner.rows[0])

}

async createPlanner(req, res) {

const { fio, email, phone } = req.body

const newPlanner = await db.query(

`INSERT INTO planner

(fio, email, phone)

values ($1, $2, $3) RETURNING \*`,

[fio, email, phone]

)

res.json(newPlanner.rows[0])

}

async updatePlanner(req, res) {

const { id, fio, email, phone } = req.body

const planner = await db.query(

`UPDATE planner set

fio = $1, email = $2, phone = $3

where id = $4 RETURNING \*`,

[fio, email, phone, id]

)

res.json(planner.rows[0])

}

async deletePlanner(req, res) {

const id = req.params.id

const companies = await db.query(

`UPDATE company

SET id\_planner = NULL

WHERE id\_planner = $1 RETURNING \*`,

[id]

)

const planner = await db.query('DELETE FROM planner where id = $1', [id])

res.json({planner: planner.rows[0], companies: companies.rows})

}

// Designer

async getAllDesigners(req, res) {

const designers = await db.query('SELECT \* FROM designer')

res.json(designers.rows)

}

async getDesigner(req, res) {

const id = rea.params.id

const designer = await db.query('SELECT \* FROM designer where id = $1', [id])

res.json(designer.rows[0])

}

async createDesigner(req, res) {

const { fio, email, phone } = req.body

const newDesigner = await db.query(

`INSERT INTO designer

(fio, email, phone)

values ($1, $2, $3) RETURNING \*`,

[fio, email, phone]

)

res.json(newDesigner.rows[0])

}

async updateDesigner(req, res) {

const { id, fio, email, phone } = req.body

const designer = await db.query(

`UPDATE designer set

fio = $1, email = $2, phone = $3

where id = $4 RETURNING \*`,

[fio, email, phone, id]

)

res.json(designer.rows[0])

}

async deleteDesigner(req, res) {

const id = req.params.id

const companies = await db.query(

`UPDATE company

SET id\_designer = NULL

WHERE id\_designer = $1 RETURNING \*`,

[id]

)

const designer = await db.query('DELETE FROM designer where id = $1', [id])

res.json({designer: designer.rows[0], companies: companies.rows})

}

}

module.exports = new Controller()