МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащаяся группы ПО-455 О.Е.Листопадова

Шифр 21

2023

# 26 Опишите этапы создания приложения

Если описать процесс разработки кратко, получится такой список:

1. Сбор требований к приложению.
2. Предварительная оценка стоимости разработки.
3. Проектирование прототипа.
4. Составление технического задания.
5. Разработка первого релиза продукта.
6. Тестирование.
7. Публикация.
8. Техническая поддержка.

Остановимся подробнее на каждом этапе.

**Сбор требований к приложению.**

Клиент звоните или оставляете заявку на сайте, или любым удобным образом. Менеджер по продажам в формате телефонного интервью собирает информацию: какое приложение необходимо сделать, для чего оно предназначено, кто будет им пользоваться, какие функции в нем будут, есть ли сайт, нужен ли сервер, будут ли интеграции с программными продуктами, например, с 1С. После сбора информации менеджер отдает заполненный бриф на оценку в отдел разработки.

Предварительная оценка стоимости разработки.

Отдел разработки оценивает, сколько времени займет создание приложения, какие специалисты будут задействованы.

Предварительная стоимость мобильного приложения может отличаться от итоговой стоимости, как в большую, так и в меньшую сторону. Точную сумму затрат на разработку приложения можно подсчитать только после составления технического задания.

**Проектирование прототипа.**

Чтобы показать, как будет работать приложение, нужен прототип - макет будущего приложения. Прототип может быть статичный или интерактивный - с кнопками действий и переходами. За дело берется аналитик. Он продумывает логику работы приложения и путь пользователя в нем - user story.  На этом этапе отдел маркетинга может предложить скорректировать идею, ориентируясь на целевую аудиторию и задачи.

Когда все готово, аналитик согласовывает с прототипы, если нужно вносит правки и передает дизайнеру. Дизайнер определяет стиль оформления приложения и основное направление, используя концепцию Material design guidelines и iOS Human Interface Guidelines с рекомендациями размеров, отступов, визуальных эффектов, анимации. Программисты также ориентируются на руководства для разработки мобильных приложений от Apple и Google.

**Составление технического задания.**

Техническое задание составляет аналитик на основе утвержденных прототипов. В техническом задании подробно описываются: функционал мобильного приложения, переходы, бизнес-процессы, основные сценарии, способы получения и передачи данных. Когда ТЗ готово, можно точно оценить стоимость проекта.

**Разработка первого релиза продукта.**

На основе ТЗ программисты выпускают первый релиз приложения. Для сложных проектов мы советуем создавать MVP - минимальную жизнеспособную версию продукта, чтобы проверить реакцию пользователей на приложение. Если необходимо, разрабатывается сервер для управления мобильным приложением и API - сервис для передачи данных между сервером и приложением.

**Тестирование.**

Приложения проверяются на ошибки с помощью Smoke Test, ручного функционального тестирования, автоматического функционального тестирования и программы Monkey для Android. Если есть ошибки, они исправляются и передается готовое приложение. Для статистики скачиваний и анализа поведения пользователей в приложении можно устанавливить Firebase Analytics.

**Публикация.**

Публикуется приложение в Google Play Market и App Store, если это мобильное приложение, либо передается исполняемый фал и при необходимости прочие файлы.

**Техническая поддержка.**

Основные задачи технической поддержки мобильного приложения:

* получение обратной связи от пользователей, решение возникающих у них затруднений;
* проверка работоспособности приложения;
* обновление для новых устройств и версий операционных систем.

# 53 Опишите функции РНР для работы с СУБД MySQL

Типичный процесс работы с СУБД в PHP-сценарии состоит из нескольких шагов:

1. Установить подключение к серверу СУБД, передав необходимые параметры: адрес, логин, пароль.
2. Убедиться, что подключение прошло успешно: сервер СУБД доступен, логин и пароль верные и так далее.
3. Сформировать правильный SQL запрос (например, на чтение данных из таблицы).
4. Убедиться, что запрос был выполнен успешно.
5. Получить результат от СУБД в виде массива из записей.
6. Использовать полученные записи в своём сценарии (например, показать их в виде таблицы).

## **Функция mysqli connect: соединение с MySQL**

Перед началом работы с данными внутри MySQL, нужно открыть соединение с сервером СУБД. В PHP это делается с помощью стандартной функции mysqli\_connect(). Функция возвращает результат — ресурс соединения. Данный ресурс используется для всех следующих операций с MySQL.

Но чтобы выполнить соединение с сервером, необходимо знать как минимум три параметра:

* Адрес сервера СУБД;
* Логин;
* Пароль.

Базовый синтаксис функции mysqli\_connect():

mysqli\_connect(<адрес сервера>, <имя пользователя>, <пароль>, <имя базы данных>);

## **Проверка соединения**

Первое, что нужно сделать после соединения с СУБД — это выполнить проверку, что оно было успешным. Эта проверка нужна, чтобы исключить ошибку при подключении к БД. Неверные параметры подключения, неправильная настройка или высокая нагрузка заставит MySQL отвергать новые подключения. Все эти ситуации приведут к невозможности соединения, поэтому программист должен проверить успешность подключения к серверу, прежде чем выполнять следующие действия.

Соединение с MySQL устанавливается один раз в сценарии, а затем используется при всех запросах к БД.

Результатом выполнения функции mysqli\_connect() будет значение специального типа — ресурс. Если подключение к MySQL не удалось, то функция mysqli\_connect() вместо ресурса вернёт логическое значение типа «ложь» — false. Хорошей практикой будет всегда проверять результат выполнения этой функции и сравнивать его с ложью.

Соединение с MySQL и проверка на ошибки:

<?php

$link = mysqli\_connect("localhost", "root", "");

if ($link == false){

print("Ошибка: Невозможно подключиться к MySQL " . mysqli\_connect\_error());

}

else {

print("Соединение установлено успешно");

}

?>

Функция mysqli\_connect\_error() просто возвращает текстовое описание последней ошибки MySQL.

## **Установка кодировки**

Первым делом после установки соединения крайне желательно явно задать кодировку, которая будет использоваться при обмене данными с MySQL. Если этого не сделать, то вместо записей со значениями, написанными кириллицей, можно получить последовательность из знаков вопроса: ?????????????????. Вызовите эту функцию сразу после успешной установки соединения: mysqli\_set\_charset($con, "utf8");

## Выполнение запросов

Следует разделять все SQL-запросы на две группы:

1. Чтение информации (SELECT).
2. Модификация (UPDATE, INSERT, DELETE).

При выполнении запросов из среды PHP, запросы из второй группы возвращают только результат их исполнения: успех или ошибку.

Запросы первой группы при успешном выполнении возвращают специальный ресурс результата. Его, в свою очередь, можно преобразовать в ассоциативный массив (если нужна одна запись) или в двумерный массив (если требуется список записей).

## **Добавление записи**

Выражение INSERT INTO используется для добавления новых записей в таблицу базы данных.

Составим корректный SQL-запрос на вставку записи с именем города, а затем выполним его путём передачи этого запроса в функцию mysqli\_query(), чтобы добавить новые данные в таблицу.

<?php

$link = mysqli\_connect("localhost", "root", "");

$sql = 'INSERT INTO cities SET name = "Санкт-Петербург"';

$result = mysqli\_query($link, $sql);

if ($result == false) {

print("Произошла ошибка при выполнении запроса");

}

Обратите внимание, что первым параметром для функции mysqli\_query() передаётся ресурс подключения, полученный от функции mysqli\_connect(), вторым параметром следует строка с SQL-запросом.

При запросах на изменение данных (не SELECT) результатом выполнения будет логическое значение — true или false, которое будет означать, что запрос выполнить не удалось. Для получения строки с описанием ошибки существует функция mysqli\_error($link).

## **Чтение записей**

Другая частая операция при работе с базами данных в PHP — это получение записей из таблиц (запросы типа SELECT). Составим SQL-запрос, который будет использовать SELECT выражение. Затем выполним этот запрос с помощью функции mysqli\_query(), чтобы получить данные из таблицы.

В этом примере показано, как вывести все существующие города из таблицы cities:

<?php

$sql = 'SELECT id, name FROM cities';

$result = mysqli\_query($link, $sql);

while ($row = mysqli\_fetch\_array($result)) {

print("Город: " . $row['name'] . "; Идентификатор: . " . $row['id'] . "<br>");

}

В примере выше результат выполнения функции mysqli\_query() сохранён в переменной $result. В этой переменной находятся не данные из таблицы, а специальный тип данных — так называемая ссылка на результаты запроса.

Чтобы получить действительные данные, то есть записи из таблицы, следует использовать другую функцию — mysqli\_fetch\_array() — и передать ей единственным параметром эту самую ссылку. Теперь каждый вызов функции mysqli\_fetch\_array() будет возвращать следующую запись из всего результирующего набора записей в виде ассоциативного массива.

Цикл while здесь используется для «прохода» по всем записям из полученного набора записей. Значение поля каждой записи можно узнать, просто обратившись по ключу этого ассоциативного массива.

## **Как получить сразу все записи в виде двумерного массива**

Иногда бывает удобно после запроса на чтение не вызывать в цикле mysqli\_fetch\_array для извлечения очередной записи по порядку, а получить их сразу все одним вызовом. PHP так тоже умеет.

Функция mysqli\_fetch\_all($res, MYSQLI\_ASSOC) вернёт двумерный массив со всеми записями из результата последнего запроса. Перепишем пример с показом существующих городов с её использованием:

<?php

$sql = 'SELECT id, name FROM cities';

$result = mysqli\_query($link, $sql);

$rows = mysqli\_fetch\_all($result, MYSQLI\_ASSOC)

foreach ($rows as $row) {

print("Город: " . $row['name'] . "; Идентификатор: . " . $row['id'] . "<br>");

}

## **Как узнать количество записей**

Часто бывает необходимо узнать, сколько всего записей вернёт выполненный SQL-запрос. Это может помочь при организации постраничной навигации или просто в качестве информации. Узнать число записей поможет функция mysqli\_num\_rows(), которой следует передать ссылку на результат запроса.

**88 «Учебное заведение»**

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server, содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.

**Решение**

Создадим форму и разместим на ней соответствующие элементы управления: поля ввода данных и кнопки для добавления, редактирования и удаления записи, а также кнопки для выполнения заданий. Внешний вид формы представлен на рисунке 1.

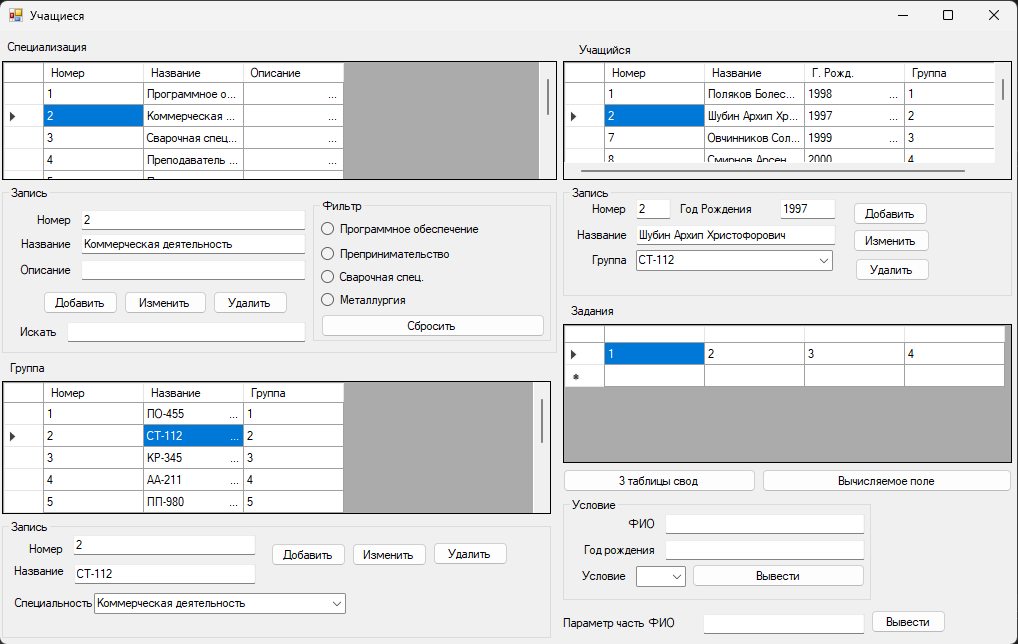


Рисунок 1 Внешний вид основной формы

Создадим вторую форму для добавления специализаций, и разместим на ней элементы управления (рисунок 2)

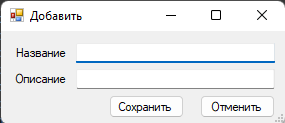


Рисунок 2 Внешний вид формы добавления специализации

Заполним таблицы данными, у нас это будут таблицы: Специализация, Студент и Группа.

На рисунке 3 представлен внешний вид приложения в котором выполнено задание вывода данных из трех таблиц таблиц.

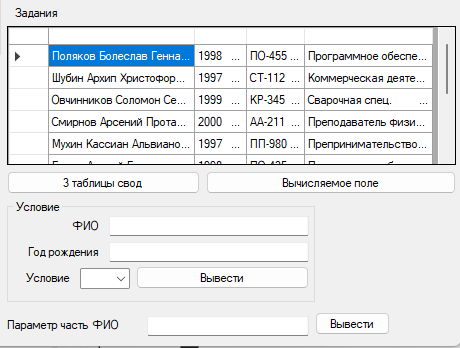


Рисунок 3 Приложение вкладка выполнения заданий

На рисунке 4 представлен результат выполнения действий над полями таблицы Студент, выведено ФИО студента, год рождения, год рождения минус 1000 и поясняющая надпись.

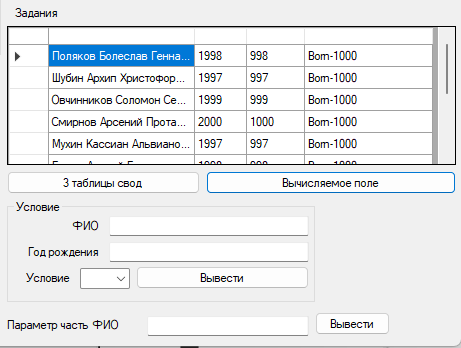


Рисунок 4 Результат выполнения задания вычисляемого поля

На рисунке 5 представлен вид приложения с выполненным заданием выбора студента по ФИО, либо по году его рождения.

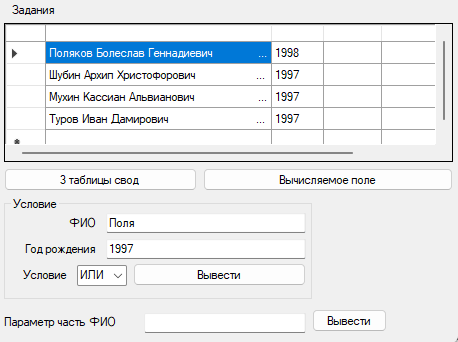


Рисунок 5 Результат выполнения задания Условие

На рисунке 6 представлен вывод задания с параметром, в качестве параметра часть ФИО студента.

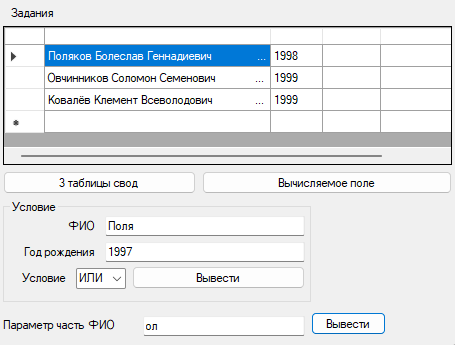
****

Рисунок 6 Результат выполнения задания Параметр

На рисунке 7 представлен результат выполнения задания Фильтр по специальности.

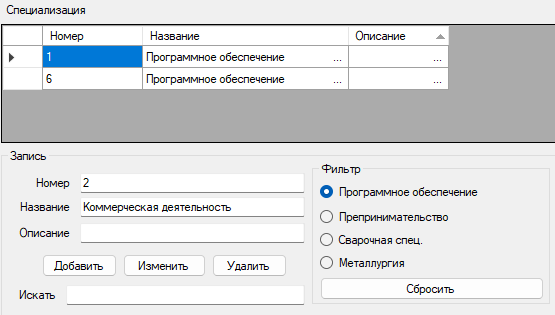
****

Рисунок 7результат выполнения задания Фильтр

На рисунке 8 представлен результат выполнения задания Поиск

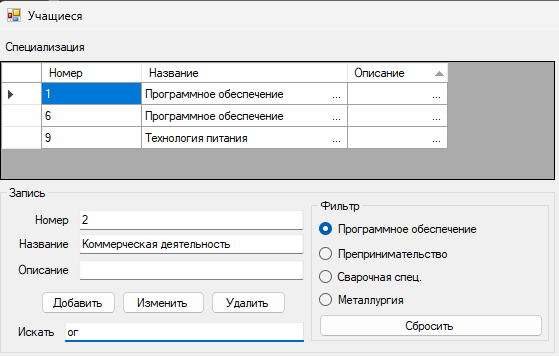


Рисунок 8результат выполнения задания Поиск

**Код программы**

**Главная Форма**

**Список использованных источников**

1 Бен-Ган, И. Microsoft SQL Server 2012. Основы T-SQL / И. Бен-Ган. М. : Эксмо, 2015. 400 с.

2 Дейт, К.Дж. Введение в системы баз данных / К.Дж. Дейт. М. : Вильямс, 2018. 1382 с.

3 Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас, И.Н. Загуменникова, П.Г. Гилевский. Минск : РИПО, 2016. 268 с.

4 Федорова, Г. Разработка и администрирование баз данных / Г. Федорова. М. : Академия, 2015. 313 с.

5 SQL справочник / К. Кляйн [и др.]. СПб. : Символ-плюс, 2016. 56 с.