WORK ORDER CONTROLLER

Getassignments:

string query = “SELECT \* FROM Assignment”;

getuserassignments:

string query = “SELECT Assignment.idAssignment, Assignment.location, Assignment.description, Assignment.date, Customer.name, Customer.address, EmployeeAssignment.idEmployeeAssignment FROM Assignment JOIN Customer ON Customer.idCustomer = Assignment.idCustomer JOIN EmployeeAssignment ON Assignment.idAssignment =EmployeeAssignment.idAssignment WHERE EmployeeAssignment.idEmployee = @0”;

getassignment:

string query = “SELECT idAssignment, location, description, date FROM Assignment WHERE idAssignment = @0”;

getworkorders/timesheet

SELECT TimesheetPart.startTime, TimesheetPart.endTime, TimesheetPart.totalTime,

TimesheetPart.description, Customer.name, Assignment.location, Employee.name

FROM TimesheetPart

JOIN EmployeeAssignment

ON TimesheetPart.idEmployeeAssignment = EmployeeAssignment.idEmployeeAssignment

JOIN Employee

ON Employee.idEmployee = EmployeeAssignment.idEmployee

JOIN Assignment

ON Assignment.idAssignment = EmployeeAssignment.idAssignment

JOIN Customer

ON Customer.idCustomer = Assignment.idCustomer

JOIN CoWorker

ON CoWorker.idTimesheetPart = TimesheetPart.idTimesheetPart

WHERE assignment.date = @0 AND Employee.idEmployee = @1;

SELECT Machine.name, Machine.number FROM Machine

JOIN workordermachine

ON workordermachine.idMachine = workordermachine.idMachine

JOIN timesheetpart

ON workordermachine.idTimesheetPart = timesheetpart.idTimesheetPart

WHERE timesheetpart.idTimesheetPart = @3;

///getstatistieken

///string query = “SELECT EmployeeAssignment.idEmployee WHERE idEmployee = @0”;

[HttpGet("getWorkType")]

public async Task<IEnumerable<string>> GetWorkType()

{

string query = "SELECT type FROM WorkType";

List<string> data = new List<string>();

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query))

while (await reader.ReadAsync())

data.Add(reader.GetString(0));

return data;

}

[HttpGet("getmachine")]

private async Task<Machine> \_GetMachine(int id)

{

return await GetMachine(id);

}

public static async Task<Machine> GetMachine(int id)

{

if (id < 0)

return null;

string query = "SELECT idMachine, type, number, name, tag, status FROM Machine WHERE idMachine=@0";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", id)))

{

await reader.ReadAsync();

return reader.HasRows ? new Machine(reader.GetInt32(0), reader.GetString(1), reader.GetInt32(2), reader.GetString(3), reader.GetString(4), reader.GetString(5)) : null;

}

}

[HttpGet("getmachines")]

public async Task<string> GetMachines()

{

string query = "SELECT name, number, tag, idMachine FROM Machine WHERE isDeleted=@0";

List<Machine> data = new List<Machine>();

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", false)))

while (reader.Read())

data.Add(new Machine(idMachine: reader.GetInt32(3), tag: reader.GetString(2), number: reader.GetInt32(1), name: reader.GetString(0)));

return JsonConvert.SerializeObject(data);

} [HttpGet("getarchiefmachine")]

public async Task<string> GetArchiefMachines()

{

string query = "SELECT name, number, tag, idMachine FROM Machine WHERE isDeleted=@0";

List<Machine> data = new List<Machine>();

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", true)))

while (reader.Read())

data.Add(new Machine(idMachine: reader.GetInt32(3), tag: reader.GetString(2), number: reader.GetInt32(1), name: reader.GetString(0)));

return JsonConvert.SerializeObject(data);

}string query = "SELECT name, number, tag, idMachine FROM Machine WHERE isDeleted=@0";

[HttpGet("addmachine/{name}/{number}/{tag}/{type}")]

public async Task<bool> AddMachine(string name, string type, int number = 0, string tag = "")

{

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

{

string query = "SELECT \* FROM Machine WHERE number=@0";

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", number)))

if (reader.HasRows)

return false;

query = "INSERT INTO Machine (name, number, tag, type, status) VALUES (@0, @1, @2, @3, @4)";

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", name),

new MySqlParameter("@1", number),

new MySqlParameter("@2", tag),

new MySqlParameter("@3", type),

new MySqlParameter("@4", "")))

return reader.RecordsAffected == 1;

}

}

Usercontroller

public static async Task<User> GetUser(string email)

{

if (string.IsNullOrWhiteSpace(email))

throw new ArgumentException();

string query = "SELECT idEmployee, name, username, role, isDeleted FROM Employee WHERE username=@0";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", email)))

{

await reader.ReadAsync();

return reader.HasRows ? new User(reader.GetInt32(0), reader.GetString(1), reader.GetString(2), reader.GetString(3), reader.GetBoolean(4)) : null;

}

}

public static async Task<User> GetUser(int id)

{

if (id < 0)

return null;

string query = "SELECT idEmployee, name, username, role, isDeleted FROM Employee WHERE idEmployee =@0";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", id)))

{

await reader.ReadAsync();

return reader.HasRows ? new User(reader.GetInt32(0), reader.GetString(1), reader.GetString(2), reader.GetString(3), reader.GetBoolean(4)) : null;

}

}

[HttpGet("register/{username}/{password}/{name}/{role}")]

public async Task<string> AddUser(string username, string password, string role)

{

if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password) || string.IsNullOrWhiteSpace(role))

return "Een van de opgegeven velden is leeg";

password = GetEncodedHash(password, "123");

if (await IsValid(username, password))//GAAT NIET WERKEN, ALS NAAM OF WACHTWOORD MAAR IETS ANDERS IS RETURNED IE FALSE

return "Gebruiker bestaat al";

string query = "INSERT INTO Employee (`username`, ‘password`, `role`) VALUES (@0, @1, @2);";

try

{

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

{

await MySqlHelper.ExecuteNonQueryAsync(conn, query,

new MySqlParameter("@0", username),

new MySqlParameter("@1", password),

new MySqlParameter("@2", role));

return "true";

}

}

catch { return "Er is iets misgegaan!"; }

}

[HttpGet("edit/{id}/{username}/{role}")]

public async Task<bool> EditUser(int id, string username, string role)

{

if (GetUser(id) == null)

return false;

string query = "UPDATE Employee SET username=@0, role=@1,idEmployee=@2";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", username),

new MySqlParameter("@1", role),

new MySqlParameter("@2", id)))

return reader.RecordsAffected == 1;

}

[HttpGet("archive/{id}")]

public async Task<bool> ArchiveUser(int id)

{

if (GetUser(id) == null)

return false;

string query = "UPDATE Employee SET isDeleted=@0 WHERE idEmployee=@1";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", true),

new MySqlParameter("@1", id)))

return reader.RecordsAffected == 1; ;

}

//[Authorize("Admin")]

public static async Task<IEnumerable<User>> GetAllUsers()

{

string query = "SELECT idEmployee, username, role, isDeleted FROM Employee WHERE isDeleted=@0";

List<User> users = new List<User>();

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", false)))

while (await reader.ReadAsync())

users.Add(new User(reader.GetInt32(0), reader.GetString(1), reader.GetString(2), reader.GetString(3), reader.GetBoolean(4)));

return users;

}

public static async Task<IEnumerable<User>> GetArchivedUsers()

{

string query = "SELECT idEmployee, name, username, role, isDeleted FROM Employee WHERE isDeleted=@0";

List<User> users = new List<User>();

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", true)))

while (await reader.ReadAsync())

users.Add(new User(reader.GetInt32(0), reader.GetString(1), reader.GetString(2), reader.GetString(3), reader.GetBoolean(4)));

return users;

}

[HttpGet("gebruikerbeheer/terughalen/{id}")]

public async Task<bool> ReAddKlant(int id)

{

if (GetUser(id) == null)

return false;

string query = "UPDATE Employee SET isDeleted=@0 WHERE idWerknemer=@1";

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", false),

new MySqlParameter("@1", id)))

return reader.RecordsAffected == 1;

}

static string GetEncodedHash(string password, string salt)

{

MD5 md5 = new MD5CryptoServiceProvider();

byte[] digest = md5.ComputeHash(Encoding.UTF8.GetBytes(password + salt));

string base64digest = Convert.ToBase64String(digest, 0, digest.Length);

return base64digest.Substring(0, base64digest.Length - 2);

}

Planningcontroller

[HttpGet("GetGebruikersWerktijden/{time}")]

public async Task<List<GebruikerTijd>> GetGebruikersWerktijden(DateTime time)

{

List<GebruikerTijd> tijden = new List<GebruikerTijd>();

List<User> users = new List<User>(await UserController.GetAllUsers());

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

{

foreach (User user in users)

{

GebruikerTijd gt = new GebruikerTijd(Employee.idEmployee, Employee.name);

tijden.Add(gt);

string query = "SELECT TimesheetPart.startTime, TimesheetPart.endTime FROM TimesheetPart JOIN EmployeeAssignment ON EmployeeAssignment.idEmployeeAssignment = TimesheetPart.idEmployeeAssignment JOIN Assignment ON Assignment.idAssignment = EmployeeAssignment.idEmployeeAssignment WHERE Assignment.date >= @0 AND Assignment.date < @1";

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", time),

new MySqlParameter("@1", time.AddDays(1))))

{

while (await reader.ReadAsync())

{

gt.vanTijd = reader["van"] as DateTime? ?? DateTime.Now;

gt.totTijd = reader["tot"] as DateTime? ?? DateTime.Now;

}

}

}

}

return tijden;

}

Opdracht Controller

[HttpPost("toevoegen")]

public async Task<bool> Toevoegen([FromBody]Opdracht opdracht)

{

using (MySqlConnection conn = await DatabaseConnection.GetConnection())

{

int opdrachtId = -1;

string query = "INSERT INTO Assignment (location, description, idCustomer, date) VALUES (@0, @1, @2, @3); SELECT LAST\_INSERT\_ID()";

using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

new MySqlParameter("@0", Assignment.location),

new MySqlParameter("@1", Assignment.description),

new MySqlParameter("@2", Assignment.customer.idCustomer),

new MySqlParameter("@3", Assignment.date)))

{

await reader.ReadAsync();

opdrachtId = reader.GetInt32(0);

}

query = "INSERT INTO EmployeeAssignment (idEmployee, idAssignment) VALUES (@0, @1)";

foreach (User user in opdracht.Users)

await MySqlHelper.ExecuteNonQueryAsync(conn, query,

new MySqlParameter("@1", idAssignment),

new MySqlParameter("@0", idEmployee));

return true;

}

}

//[HttpGet("alle/{archive}")]

//public async Task<IEnumerable<Opdracht>> GetOpdrachten(bool archived)

//{

// string query = "SELECT Opdracht.\*, COUNT(OpdrachtWerknemer.idOpdrachtWerknemer) as count FROM Opdracht LEFT JOIN OpdrachtWerknemer ON Opdracht.idOpdracht = OpdrachtWerknemer.idOpdracht GROUP BY Opdracht.idOpdracht";

// List<Opdracht> opdrachten = new List<Opdracht>();

// using (MySqlConnection conn = await DatabaseConnection.GetConnection())

// using (MySqlDataReader reader = await MySqlHelper.ExecuteReaderAsync(conn, query,

// new MySqlParameter("@0", 1)))

// while (await reader.ReadAsync())

// opdrachten.Add(new Opdracht(

// idOpdracht: reader["idOpdracht"] as int? ?? -1,

// locatie: reader["locatie"] as string,

// beschrijving: reader["beschrijving"] as string,

// datum: reader["datum"] as DateTime? ?? null)

// {klant = await WerkbonController.GetKlant(reader["idKlant"] as int? ?? -1),

// gebruikerCount = (int)(reader["count"] as long? ?? (long)0)

// });

// return opdrachten;

//} }