ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет по компютърни системи и управление

Бази от Данни

Студент : *Георги Челенков, ФКСУ , 45 гр*

Фак. № : 121213099

Курсова работа

Система за сервиз на електронни компоненти

Приложението е изградено с помощта на следните **JAVA** класове:

- ComponentService
- DatePicker
- MySQLConnect
- ObservingTextField
- ResultSetTable

В **ComponentService** класа с помощта на **Swing** библиотеката е изградена клиентската версия на приложението чрез богат набор от компоненти и интеракции. Съставен е и многонишков интерфейс, който позволява използване на приложението от множество клиенти. В отделна нишка се отварят и създават инстанции на класа MySQLConnect, който контролира взаимодействието на приложението с базата от данни component_service, имащ свойствата на сървър за конкретното приложение.

Чрез следния код е показа структурата на класа:

```
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				window = new ComponentService();
				databaseConnection = new MySQLConnect();
				databaseConnection.connectToDatabase();
				window.frame.setVisible(true);

window.checkDatabaseConnectivity(databaseConnection.hasConnected());
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
	}
```

MySQLConnect класа реализира сървърната част на приложението. Той се грижи за осъществяване на връзката на приложението с базата от данни както и за реализиране на заявките към същата.

DatePicker класа се използва за реализиране на календара в приложението.

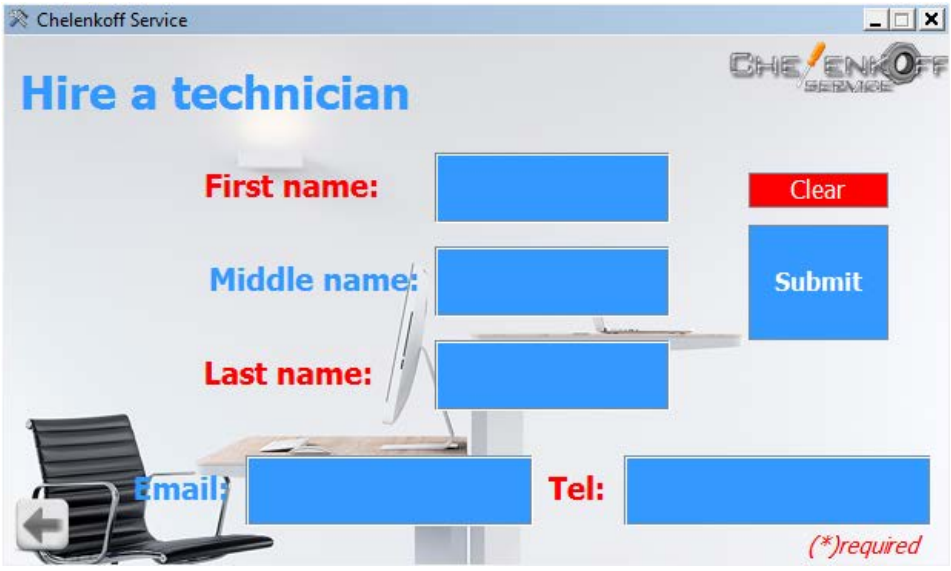
ResultSetTable класа се използва за обработка на result set-а получен при изпращане на заявка до базата от данни.

ObservingTextField класа се използва за разширение на функционалността на обикновения text field с цел – по – елегантната обработка на полета от тип “Date”.

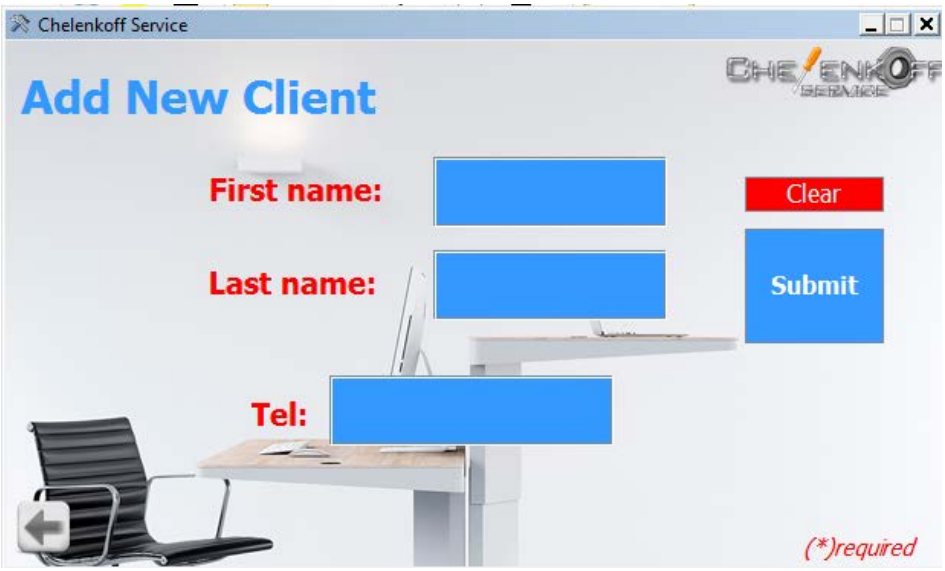
Функционалност и дизайн на програмата:

Това е главното меню на програмата. То се състои от два интерактивни бутона, които отвеждат към две менюта – това, което се използва от офис – служителите и това, порзвано от специалистите.

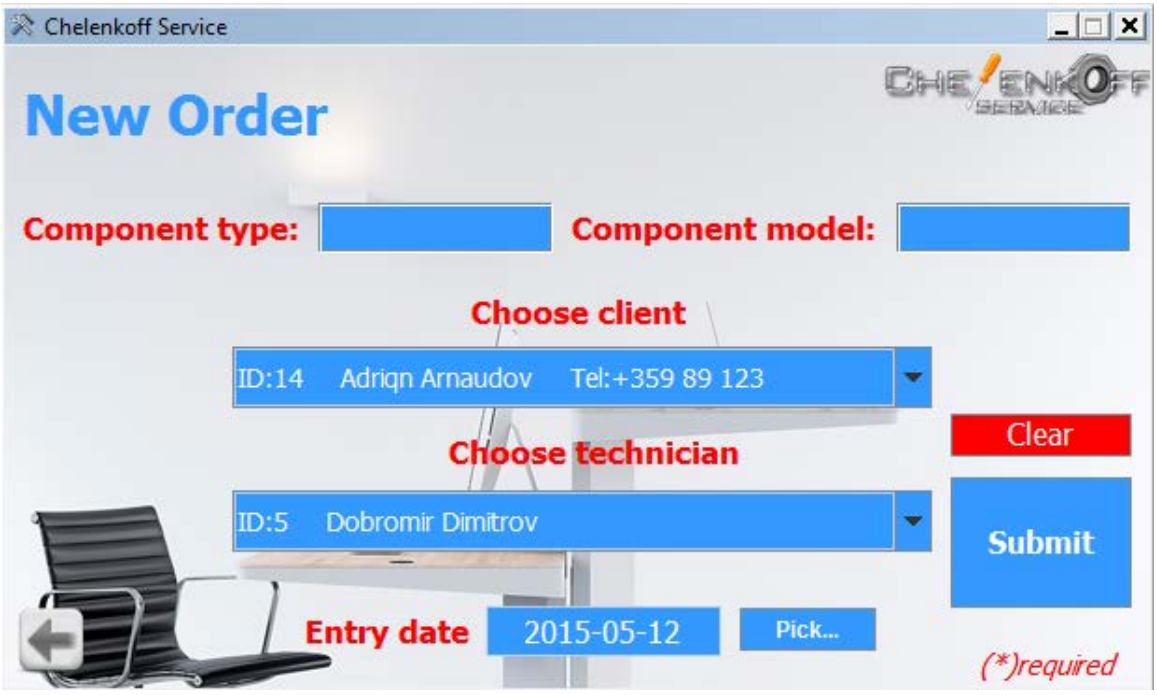
Това е менюто ползвано от офис – служителите. То има следната функционалност – Добавяне на нова заявка, Добавяне на нов клиент и Наемане на техник.



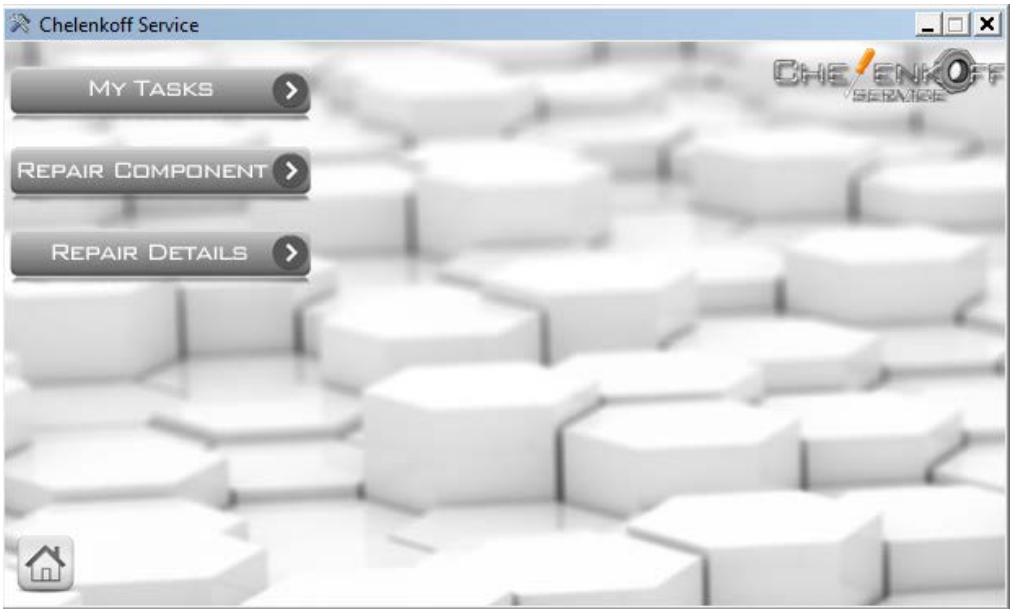
Менюто използвано за наемане на нов техник. Натискането на бутона “Submit” води до изпълнение на INSERT заявка към базата от данни (заявката е показана по-надолу).



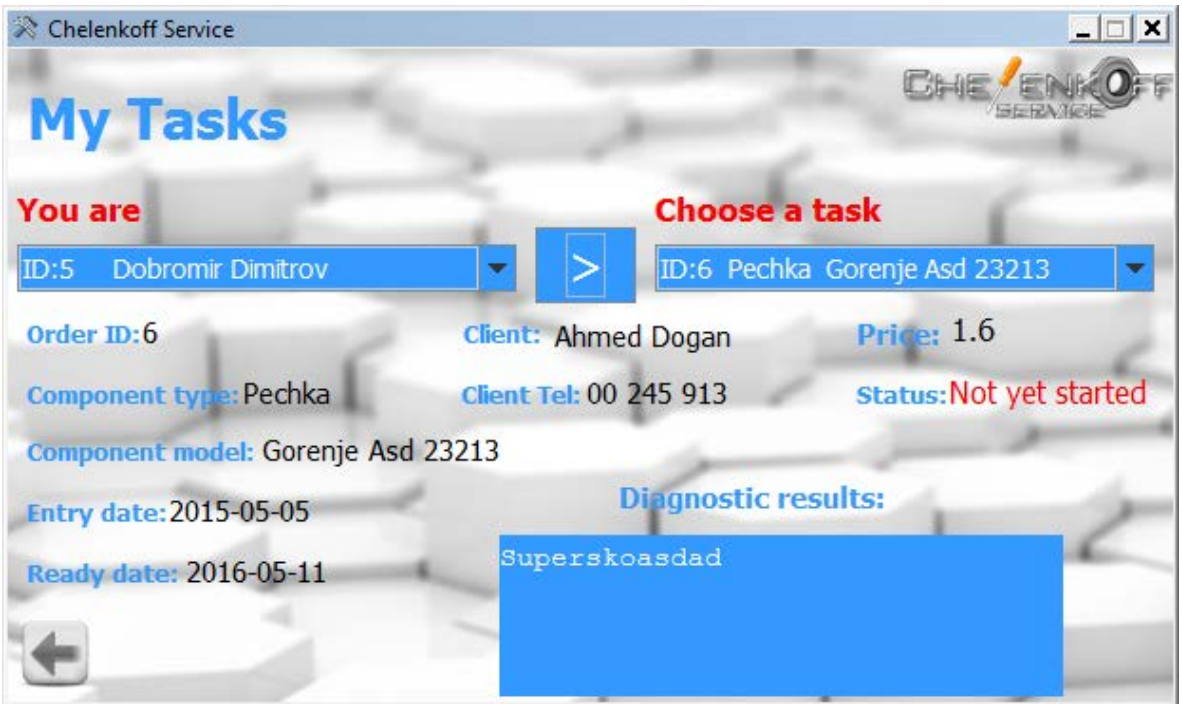
Менюто използвано за добавяне на нови клиенти. Натискането на бутона “Submit” води до изпълнение на INSERT заявка към базата от данни (заявката е показана по-надолу).



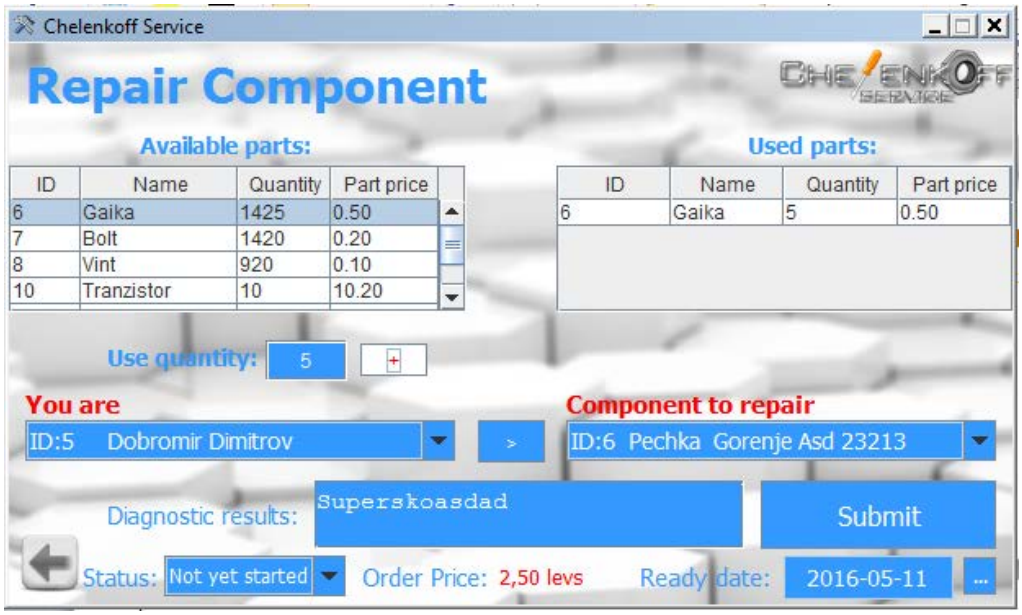
Менюто използвано за създаване на нова поръчка. Натискането на бутона “Submit” води до изпълнение на INSERT заявка към базата от данни (заявката е показана по-надолу). Бутонът “Pick…” е реализиран посредством класа “DatePicker” и води до показване на интерактивен календар. Съответните ComboBox-ове са попълни чрез SELECT заявки, чиито код е даден по – надолу.



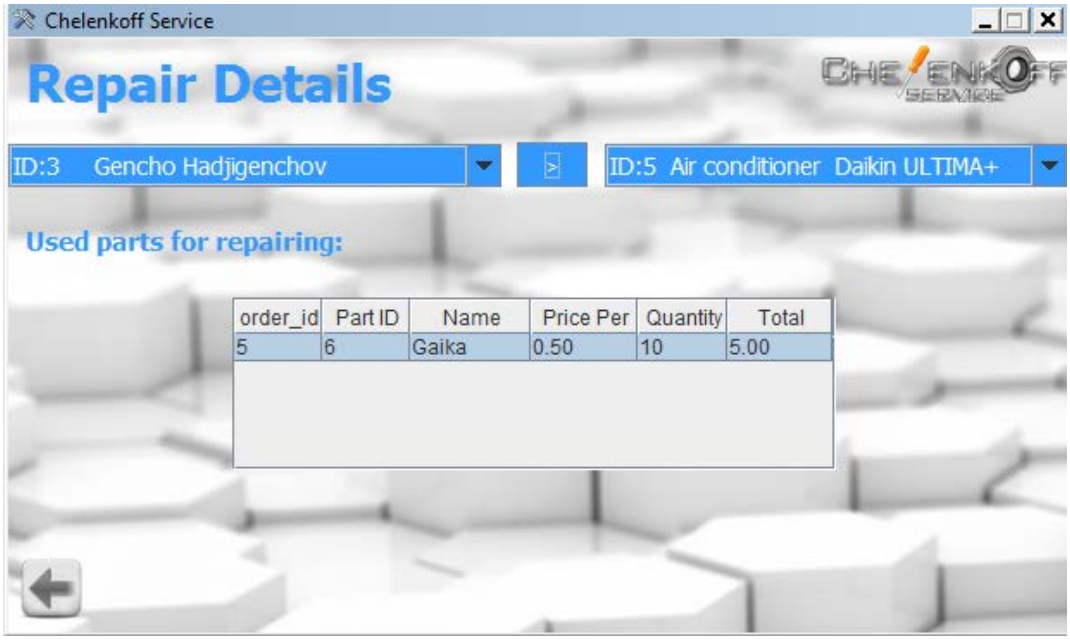
Това е менюто ползвано от техниците. То има следната функционалност – Проверка на възложените задания на даден техник, Панел, в който може да се въвежда информация относно ремонта на даден компонент и Панел, показващ детайлна информация за вложените в ремонта на даден компонент части.



Меню, в което при избор на съответен техник се визуализират в “ComboBox” неговите задачи. При избор на задачата, се визуализира информация относно ремонтните дейности на компонента.



Панел, в който протича процеса на ремонт на даден компонент. Визуализира се таблица със всички налични части в склада, както и таблица , в която се отбелязват вложените части. Показана е и кратка информация относно ремонта – резултати от диагностиката, цена на ремонта, статус, дата на приключване на ремонта.



Показана е детайлна информация относно вложените в ремонта на даден компонент части, тяхното название, партиден номер, количество, единична цена и обща цена.

CREATE заявки

```
CREATE DATABASE component_service;
```

```
CREATE TABLE technicians(
	technic_id SMALLINT(4) UNSIGNED AUTO_INCREMENT
PRIMARY KEY,
	first_name VARCHAR(50) NOT NULL,
	middle_name VARCHAR(50) NULL,
	last_name VARCHAR(50) NOT NULL,
	is_available ENUM('Y','N') NOT NULL DEFAULT 'Y',
	tel_num VARCHAR(30) NOT NULL,
	email VARCHAR(50) NULL,
	num_of_repaired SMALLINT UNSIGNED NULL,
	num_of_being_repaired TINYINT(2) UNSIGNED NULL
	)ENGINE = InnoDB;
```

```
CREATE TABLE clients(
	client_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
	first_name VARCHAR(50) NOT NULL,
	last_name VARCHAR(50) NOT NULL,
	tel_num VARCHAR(30) NOT NULL
	)ENGINE = InnoDB;
```

```
CREATE TABLE components(
	order_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
	type VARCHAR(255) NOT NULL,
	model VARCHAR(255) NOT NULL,
	order_price FLOAT(7,2) UNSIGNED NULL,
	diagnostic_results TEXT NULL,
	status ENUM('not_started','in_progress','ready') NOT NULL DEFAULT 'not_started',
	entry_date DATE    NOT NULL,
	ready_date DATE NULL,
	client_id INT UNSIGNED NOT NULL,
	FOREIGN KEY(client_id)
		REFERENCES clients(client_id)
		ON DELETE CASCADE,
	technic_id SMALLINT(4) UNSIGNED NULL,
	FOREIGN KEY (technic_id)
		REFERENCES technicians(technic_id)
		ON DELETE SET NULL
		)ENGINE = InnoDB;
```

```
CREATE TABLE parts(
        part_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        quantity SMALLINT UNSIGNED NULL,
        price_per FLOAT(7,2) UNSIGNED NULL
        )ENGINE = InnoDB;

CREATE TABLE replaced_components_parts(
        comp_id INT UNSIGNED NOT NULL,
        part_id INT UNSIGNED NOT NULL,
        qty SMALLINT UNSIGNED NOT NULL,
        UNIQUE(comp_id,part_id),
        FOREIGN KEY(comp_id)
                REFERENCES components(order_id)
                ON DELETE CASCADE,
        FOREIGN KEY(part_id)
                REFERENCES parts(part_id)
                ON DELETE CASCADE
        )ENGINE = InnoDB;
```

ПРОЦЕДУРИ

```
DELIMITER |
CREATE PROCEDURE add_technician(IN first VARCHAR(50),IN middle VARCHAR(50),IN last
VARCHAR(50), IN tel VARCHAR(30),IN email VARCHAR(50)  )
        BEGIN
                IF first IS NULL OR first = ' '
                        THEN SELECT "Invalid first name!" AS result;
                        ELSE
                                IF last IS NULL OR last = ' '
                                        THEN SELECT "Invalid last name!" AS result;
                                        ELSE
                                                IF tel IS NULL OR tel = ' '
                                                        THEN SELECT "Invalid tel number!" AS result;
                                                        ELSE
                                                                IF EXISTS(SELECT * FROM technicians
WHERE first_name = first AND last_name = last AND tel_num = tel)
```

```
THEN SELECT "Technician already exists!" AS result, "His Personal number is:" AS statement,
                                                                    technic_id AS id
FROM technicians WHERE technicians.first_name = first AND

                                                technicians.middle_name = middle AND

                                                technicians.last_name = last AND

                                                technicians.tel_num = tel AND

                                                technicians.email = email;
                                                        ELSE
                                                                INSERT    INTO    technicians
(first_name,middle_name,last_name,tel_num,email)

        VALUES(first,middle,last,tel,email);
                        SELECT "Technician added successfully!" AS result,"His personal number
is:" AS statement,
technic_id AS id FROM technicians WHERE technicians.first_name = first AND

                                                technicians.middle_name = middle AND
                                        technicians.last_name = last AND
                                                        technicians.tel_num = tel AND
                                        technicians.email = email;
                                                END IF;
                                        END IF;
                                END IF;
                        END IF;
        END |
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE add_client(IN first VARCHAR(50),IN last VARCHAR(50),
                                                    IN tel VARCHAR(30))
        BEGIN
                IF first IS NULL OR first = ' '
                        THEN SELECT "Invalid first name!" AS result;
                        ELSE
                                IF last IS NULL OR last = ' '
                                        THEN SELECT "Invalid last name!" AS result;
                                        ELSE
                                                IF tel IS NULL OR tel = ' '
                                                        THEN SELECT "Invalid tel number!" AS result;
                                                        ELSE
                                                                IF EXISTS(SELECT * FROM clients WHERE
first_name = first AND last_name = last AND

                                        tel_num = tel)
                                                                        THEN SELECT "Client already
exists!" AS result, "His Personal number is:" AS statement,
                                                                                client_id AS id
FROM clients WHERE clients.first_name = first AND

                                                        clients.last_name = last AND

                                                        clients.tel_num = tel;
                                                                ELSE
                                                                        INSERT INTO clients
(first_name,last_name,tel_num)
                                                                        VALUES(first,last,tel);
                                                                        SELECT "Client added
successfully!" AS result,"His personal number is:" AS statement
                                                                        ,client_id AS id FROM clients
WHERE clients.first_name = first AND

                                                        clients.last_name = last AND

                                                        clients.tel_num = tel;
                                                                END IF;
                                                END IF;
                                END IF;
                END IF;
        END |
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE create_order(IN comp_type VARCHAR(255),IN comp_model VARCHAR(255), IN client_id INT, IN technic_id SMALLINT(4), IN entry_date DATE)
	BEGIN
		IF comp_type IS NULL OR comp_type = ' ' OR comp_type = ''
			THEN SELECT "Invalid component type!" AS result;
			ELSE
				IF comp_model IS NULL OR comp_model = ' ' OR comp_model = ''
					THEN SELECT "Invalid component model!" AS result;
					ELSE
						IF entry_date IS NULL OR entry_date = ' ' OR entry_date ='' THEN SELECT "Invalid entry date!" AS result;
							ELSE IF EXISTS(SELECT * FROM components WHERE components.type = comp_type AND components.model = comp_model AND components.technic_id = technic_id AND components.client_id = client_id AND
components.entry_date = entry_date)
THEN SELECT CONCAT("This component is already being repaired!
Order number is: ",(SELECT order_id FROM components WHERE
components.type = comp_type AND
components.model = comp_model AND
components.technic_id = technic_id AND

components.client_id = client_id AND

components.entry_date = entry_date)) AS result;
ELSE
INSERT INTO components (type,model,entry_date,client_id,technic_id)
		VALUES(comp_type,comp_model,entry_date,client_id,technic_id);
		SELECT CONCAT("Component prepared for repairing successfully!
			Order number is: ",(SELECT order_id FROM components WHERE
components.type = comp_type AND
components.model = comp_model AND
components.technic_id = technic_id AND
components.client_id = client_id AND
components.entry_date = entry_date)) AS result;
END IF;
END IF;
END IF;
		END IF;
	END |
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE component_info(IN id INT)
	BEGIN
		SELECT components.order_id AS id, components.type AS type, components.model AS model,
			components.entry_date AS entry, components.ready_date AS ready, clients.first_name AS first_name,
			clients.last_name AS last_name, clients.tel_num AS tel, components.status AS status,
			components.order_price AS price, components.diagnostic_results AS result FROM components
				JOIN clients
					ON components.client_id = clients.client_id
			WHERE components.order_id = id;
	END
	|
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE component_parts_info(IN id INT)
	BEGIN
		SELECT
			components.ready_date AS ready,
			components.status AS status,
			components.order_price AS price,
			components.diagnostic_results AS result
			FROM components
				WHERE components.order_id = id;
	END
	|
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE repair_component(IN comp_id INT, IN part_id INT, IN qty SMALLINT)
	BEGIN

		SET @qty_entered = qty;
		SET @part_id_entered = part_id;

		UPDATE replaced_components_parts SET replaced_components_parts.qty = replaced_components_parts.qty + qty
			WHERE replaced_components_parts.part_id = part_id AND replaced_components_parts.comp_id = comp_id;
		IF (ROW_COUNT() = 0)
			THEN
				INSERT INTO replaced_components_parts(comp_id,part_id,qty)
					VALUES(comp_id,part_id,qty);
				SELECT "Successfully inserted!";
			ELSE
				SELECT "Successfully updated!";
		END IF;


	END
	|
DELIMITER ;
```

```
DELIMITER |
CREATE PROCEDURE update_component_info(IN comp_id INT,IN diagnostic_results TEXT, IN status VARCHAR(50),
                                       IN order_price FLOAT(7,2), IN ready_date DATE)
	BEGIN

		UPDATE components SET components.order_price = order_price
			WHERE components.order_id = comp_id;

		UPDATE components SET components.diagnostic_results = diagnostic_results
			WHERE components.order_id = comp_id;

		UPDATE components SET components.status = status
			WHERE components.order_id = comp_id;

		UPDATE components SET components.ready_date = ready_date
			WHERE components.order_id = comp_id;


	END
	|
DELIMITER ;
```

ТРИГЕРИ

```
DELIMITER //
CREATE TRIGGER qtyUpdate
AFTER UPDATE ON replaced_components_parts FOR EACH ROW
 BEGIN
  UPDATE parts SET parts.quantity = parts.quantity - @qty_entered
            WHERE parts.part_id = @part_id_entered;
 END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER qtyInsert
AFTER INSERT ON replaced_components_parts FOR EACH ROW
 BEGIN
  UPDATE parts SET parts.quantity = parts.quantity - @qty_entered
            WHERE parts.part_id = @part_id_entered;
 END;//
DELIMITER ;
```

Изгледи

```
CREATE VIEW repair_details
      AS SELECT components.order_id AS 'order_id',parts.part_id AS 'Part ID', parts.name AS 'Name',parts.price_per AS 'Price Per',replaced_components_parts.qty AS 'Quantity', (replaced_components_parts.qty * parts.price_per) AS 'Total'
            FROM components JOIN replaced_components_parts ON replaced_components_parts.comp_id = components.order_id
                                    JOIN parts ON parts.part_id = replaced_components_parts.part_id
            ORDER BY components.order_id,parts.part_id,parts.name;
```