**Курсов проект по**

**„Web програмиране“**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Име, Презиме,Фамилия | Фак №: | Роля |
| 1. | Василена Йорданова Даскалска | 22621660 | Разработка на софтуера, тестване |

Срок за предаване: до края на семестъра

Водещ преподавател:

/доц. В. Божикова/

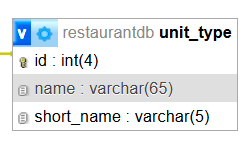
1. **Задание за разработка**
   1. **Основна цел:**

Да се изгради динамична система за управление на ресторантско меню, съставки и поръчки с възможност за лесно актуализиране, справки и автоматично обработване на взаимосвързани данни (напр. при изтриване на записи).

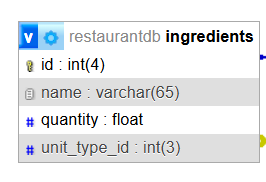
* 1. **Функционални изисквания:**
     1. **Управление на съставките:**
* Възможност за добавяне на нови съставки с имена, количества и единици.
* Редактиране на съществуващи съставки.
* Изтриване на съставки с автоматично премахване на свързаните данни от таблицата **dish\_ingredients**.
  + 1. **Управление на ястия**
* Добавяне на ястия с описание и съставки.
* Редактиране на информацията за ястията (име, описание, съставки и количества).
* Изтриване на ястия с автоматично премахване на съответните записи от **dish\_ingredients** и проверка дали има запис в **order\_dishes**. Ако има запис да не позволява да се изтрие.
  + 1. **Управление на поръчки**
* Създаване на нови поръчки, свързани с клиенти и ястия.
* Изчисляване на използваните съставки на база направените поръчки.
* Редактиране на информацията за поръчка.
* Изтриване на поръчка и автоматично премахване на свързаните в поръчки ястия, ако съответните записи в **orders** бъдат изтрити.
  + 1. **Справки и отчети**
* Генериране на справка за наличните съставки:
* Показване на оставащи количества.
* Изчисляване на липсващи съставки (недостатъчни наличности).
* Отчитане само на използвани съставки в активни поръчки.
  + 1. **Връзки и зависимости**
* Автоматично управление на връзките между таблиците с **ON DELETE CASCADE** за:
* **dish\_ingredients** при триене на ястия.
* **order\_dishes** при триене на поръчки.
  + 1. **Визуализация и интерфейс**
* Потребителски интерфейс за управление на таблиците чрез:
* Форми за въвеждане и редактиране.
* Таблици за визуализация на данните.
* Бутон за изтриване на записи с потвърждение от потребителя.
  1. **Нефункционални изисквания**
     1. **Обработка на данни:**
* За всяка от таблиците в базата данни трябва да бъде осигурена унифицирана форма за стандартни операции като: въвеждане, корекции, изтриване и търсене на данни.
  + 1. **Скорост на работа:**
* Системата трябва да отговаря на заявки за справки и операции с базата данни в рамките на до 2 секунди за до 10,000 записа.
  + 1. **Съвместимост:**
* Приложението трябва да бъде съвместимо с основните съвременни браузъри: Google Chrome, Mozilla Firefox, Microsoft Edge и Safari.
* Интерфейсът трябва да бъде адаптивен за работа на устройства с различни размери на екрана (настолни компютри, таблети и смартфони).
  + 1. **Сигурност:**
* Системата трябва да валидира всички входни данни, за да предотврати SQL инжекции и други видове атаки.
  + 1. **Поддръжка на множество езици:**
* Системата трябва да има възможност за лесна локализация и добавяне на нови езици чрез използване на външни файлове за превод.
  + 1. **Надеждност:**
* Системата трябва да предотвратява загуба на данни при прекъсване на връзката с базата данни.
* Да се използват транзакции за гарантиране на консистентност на данните при сложни операции.
  + 1. **Разширяемост:**
* Архитектурата на системата трябва да позволява лесно добавяне на нови функционалности или промяна на съществуващите.
  + 1. **Поддръжка и помощ:**
* Системата трябва да разполага с помощен модул или документация за потребителите, описваща как да се използват основните функционалности.

1. **Реализация на проекта**
   1. **Проектиране:**
      1. **Концептуален и логически дизайн на базата данни (БД)**

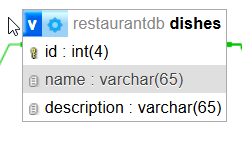
* Създадена е структура на базата данни, включваща следните основни таблици:
* **unit\_type**: за съхранение на информация за вида на мерните единици.
* **id**: Съхранява номера на въведената мерна единица. Той е първичен ключ и се инкрементира автоматично.
* **name:** Съхранява името на мерната единица. (напр. „Килограм“).
* **short\_name:** Съхранява краткото име на мерната единица. (напр. „Кг.“)
* **Структура на таблицата:**



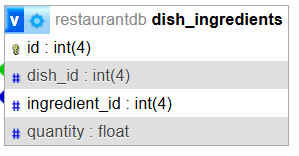
* **ingredients:** за съхранение на информация за съставките.
* **id:** Съхранява номера на въведената съставка. Той е първичен ключ и се инкрементира автоматично.
* **name:** Съхранява името на съставката. (напр. „Яйца“)
* **quantity:** Съхранява количеството на съставката. (напр. „150“)
* **unit\_type\_id:** Съхранява номера на мерната единица. Това е външен (вторичен) ключ към таблицата **unit\_type.**
* **Структура на таблицата:**



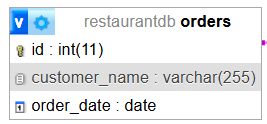
* **dishes:** за съхранение на данни за ястията.
* **id:** Съхранява номера на въведеното ястие. Той е първичен ключ и се инкрементира автоматично.
* **name:** Съхранява името на ястието. (напр. „Крем карамел“)
* **description**: Съхранява описание на ястието. (напр. „Вкусен сладкиш от яйца, прясно мляко и захар“)
* **Структура на таблицата**:



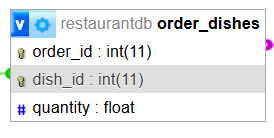
* **dish\_ingredients**: за връзка между ястията и използваните съставки.
* **id**: Съхранява номера на въведеното ястие със съставката за него. Той е първичен ключ и се инкрементира автоматично.
* **dish\_id**: Съхранява номера на въведеното ястие. Това е външен (вторичен) ключ към таблицата **dishes.**
* **ingredient\_id:** Съхранява номера на въведената съставка. Това е външен (вторичен) ключ към таблицата **ingredients.**
* **quantity:** Съхранява количеството на съставката използвана за ястието. (Напр. 150)
* **Структура на таблицата:**



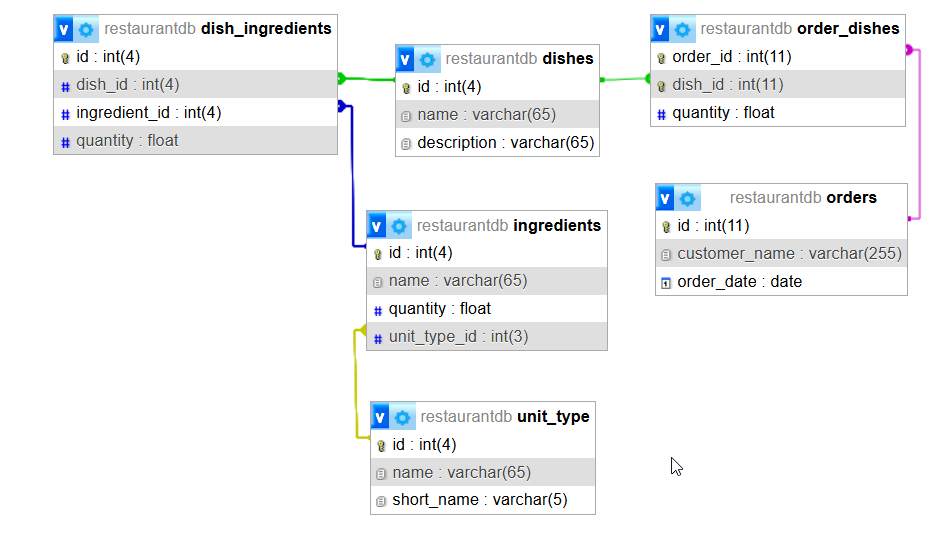
* **orders**: за съхранение на данни за клиентските поръчки.
* **id:** Съхранява номера на въведената поръчка. Той е първичен ключ и се инкрементира автоматично.
* **customer\_name:** Съхранява името на клиента. (Напр. „Василена Даскалска“)
* **order\_date:** Съхранява датата на създаване на поръчката. (Напр. „12/12/2024“)
* **Структура на таблицата:**



* **dish\_orders:** за връзка между поръчките и ястията в тях.
* **order\_id:** Съхранява номера на въведената поръчка. Това е външен (вторичен) ключ към таблицата **orders.**
* **dish\_id:** Съхранява номера на въведеното ястие. Това е външен (вторичен) ключ към таблицата **dishes.**
* **quantity:** Съхранява количеството на поръчаното ястие. **(Напр. 2)**
* **Структура на таблицата:**



* Връзките между таблиците са реализирани с помощта на външни ключове **(FOREIGN KEY)** с политики за автоматично изтриване **(ON DELETE CASCADE).**
  + 1. **ERD диаграма:**



* 1. **Програмиране:**
     1. **Модули и функционалности:**
* Създаване, редактиране и изтриване на записи в таблиците с данни (CRUD операции).
* Те се случват, чрез поп ъп прозорци при натискане на бутона „**Добави**“, „**Редактирай**“ или „**Изтрий**“.
* Логиката е обвързана с JavaScript за показване на прозорците за добавяне или редакция и затварянето им. Също така и за прихващане на избрания от потребителя ред от таблицата.
* Тъй като редакцията и изтриването стават от реда в таблицата няма нужда да потребителя да пише ръчно, кой запис иска да изтрие или редактира.
* Използва се JavaScript и за автоматично обновяване на таблиците в реално време след редакция, добавяне и изтриване.
* Код с обяснение на логиката за добавяне и редакция. (Всеки друг работи по подобен начин)
  + - * + **add\_unit.php:** За добавяне на мерна единица

<?php

**// Включване на конфигурационния файл, съдържащ връзката с базата данни**

include "config.php";

**// Проверка дали заявката към скрипта е изпратена** **чрез метода POST**

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Вземане на стойностите на полетата "name" и "short\_name" от изпратената форма

$name = $\_POST['name'];

$short\_name = $\_POST['short\_name'];

**// Проверка дали всички полета са попълнени**

if (!$name || !$short\_name) {

**// Извеждане на съобщение за грешка**

echo "Всички полета са задължителни!";

exit; **// Прекратяване на изпълнението на скрипта**

}

**// Подготовка на SQL заявка за добавяне на нова мерна единица**

$query = "INSERT INTO unit\_type (name, short\_name) VALUES (?, ?)";

**// Подготовка на SQL заявката за безопасно изпълнение (избягване на SQL инжекции)**

$stmt = $dbConn->prepare($query);

**// Свързване на стойностите на променливите с параметрите на заявката**

$stmt->bind\_param("ss", $name, $short\_name);

**// Изпълнение на заявката**

if ($stmt->execute()) {

**// Ако заявката е изпълнена успешно, извеждане на съобщение за успех**

echo "Новата мерна единица е успешно добавена!";

} else {

**// Ако има грешка при изпълнението, извеждане на съобщение за грешка**

echo "Грешка при добавянето: " . $stmt->error;

}

**// Затваряне на подготвеното изявление**

$stmt->close();

**// Затваряне на връзката с базата данни**

$dbConn->close();

}

* + - * + **update\_unit.php**

<?php

**// Включване на конфигурационния файл за връзка с базата данни**

include "config.php";

**// Проверка дали заявката е направена чрез POST метод**

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

**// Получаване на стойностите от POST данните**

**// ID на запис за обновяване**

$id = $\_POST['id'];

**// Ново име на мерната единица**

$name = $\_POST['name'];

**// Ново кратко име на мерната единица**

$short\_name = $\_POST['short\_name'];

**// Проверка дали всички задължителни полета са попълнени**

if (!$id || !$name || !$short\_name) {

**// Ако някое поле липсва, извежда съобщение за грешка и спира изпълнението**

echo "Всички полета са задължителни!";

exit;

}

**// Подготовка на SQL заявката за обновяване на запис**

$query = "UPDATE unit\_type SET name = ?, short\_name = ? WHERE id = ?";

**// Подготвяне на заявката за защита от SQL инжекция**

$stmt = $dbConn->prepare($query);

**// Свързване на параметрите към подготвената заявка**

**// "ss" означава, че първите два параметъра са текстови, а "i" означава, че третият е цяло число**

$stmt->bind\_param("ssi", $name, $short\_name, $id);

**// Изпълнение на заявката**

if ($stmt->execute()) {

**// Ако изпълнението е успешно, се извежда съобщение за успех**

echo "Записът е успешно обновен!";

} else {

**// Ако възникне грешка, се извежда съобщение за грешка**

echo "Грешка при обновяването: " . $stmt->error;

}

**// Затваряне на подготвената заявка**

$stmt->close();

**// Затваряне на връзката с базата данни**

$dbConn->close();

}

?>

* + - * + JavaScript функции в Measures.php

**// Отваряне на формата за добавяне**

**function** **openAddForm()**

{

**// Показва контейнера на формата за добавяне, като променя стила му на "block"** document.getElementById('formContainer').style.display = 'block';

**// Показва наслагващия се слой (overlay), като променя стила му на "block"** document.getElementById('overlay').style.display = 'block'; **// Задава заглавие на формата - "Добавяне на нова мерна единица"** document.getElementById('formTitle').textContent = 'Добавяне на нова мерна единица';

**// Изчиства скритото поле за ID, ако преди това е било попълнено (използва се за редактиране)** document.getElementById('formId').value = ' ';

**// Изчиства текстовото поле за име на мерната единица**

document.getElementById('formName').value = ' ';

**// Изчиства текстовото поле за кратко име на мерната единица** document.getElementById('formShortName').value = ' ';

}

**// Функция за отваряне на формата за редакция**

**function editRow(id, name, shortName)**

{

**// Показване на контейнера за формата**

document.getElementById('formContainer').style.display = 'block';

**// Показване на слоя за затъмнение (overlay) зад формата**

document.getElementById('overlay').style.display = 'block';

**// Промяна на заглавието на формата, за да указва, че това е режим за редакция**

document.getElementById('formTitle').textContent = 'Редакция на мерна единица';

**// Попълване на скритото поле за ID на реда, който ще се редактира**

document.getElementById('formId').value = id;

**// Задаване на стойност на полето за име с текущото име на избраната мерна единица**

document.getElementById('formName').value = name;

**// Задаване на стойност на полето за кратко име с текущото кратко име на избраната мерна единица**

document.getElementById('formShortName').value = shortName;

}

**// Функция за затваряне на формата**

**function closeForm()** {

**// Скрива контейнера на формата, за да спре да се вижда на екрана**

document.getElementById('formContainer').style.display = 'none';

**// Скрива слоя за затъмнение (overlay), който е зад формата**

document.getElementById('overlay').style.display = 'none';

}

**// Функция за обработка на формата**

**function handleFormSubmit()** {

**// Извличане на стойността на ID от скритото поле на формата (използва се при редакция)**

const id = document.getElementById('formId').value;

**// Извличане на въведеното име на мерната единица**

const name = document.getElementById('formName').value;

**// Извличане на въведеното кратко име на мерната единица**

const shortName = document.getElementById('formShortName').value;

**// Определяне на URL-то за заявката: ако има ID, това е редакция (update\_unit.php), ако не, добавяне** (add\_unit.php)

const url = id ? 'update\_unit.php' : 'add\_unit.php';

**// Създаване на данни за заявката: ако има ID, включва ID, име и кратко име, иначе само име и кратко име**

const data = id ? `id=${id}&name=${name}&short\_name=${shortName}` : `name=${name}&short\_name=${shortName}`;

**// Изпращане на заявката към сървъра чрез `fetch`**

fetch(url, {

method: 'POST', **// Методът на заявката е POST**

headers: { 'Content-Type': 'application/x-www-form-urlencoded' }, **// Установяване на правилния формат на данните**

body: data **// Данните, които ще бъдат изпратени към сървъра**

})

.then(response => response.text()) **// Преобразуване на отговора от сървъра в текст**

.then(data => {

**// Показване на съобщение с отговора от сървъра (например успех или грешка)**

alert(data);

**// Затваряне на формата, след като обработката е завършена**

closeForm();

**// Презареждане на страницата, за да се покажат актуализациите**

location.reload();

})

.catch(error => console.error('Error:', error)); **// Логване на грешки, ако нещо се обърка**

**// Връща false, за да предотврати стандартното поведение на формата (презареждане на страницата)**

return false;

}

* Код с обяснение на логиката за изтриване. (Всеки друг работи по подобен начин)
  + - * + **delete\_unit\_type.php**

<?php

**// Включване на конфигурационния файл за връзка с базата данни**

include "config.php";

**// Проверка дали заявката е от тип POST**

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

**// Извличане и валидиране на ID от POST данните, като се уверяваме, че е цяло число**

$id = isset($\_POST['id']) ? intval($\_POST['id']) : 0;

**// Проверка дали ID-то е валидно (по-голямо от 0)**

if ($id > 0) {

**// Подготвяне на SQL заявка за изтриване на запис с определено ID**

$query = "DELETE FROM unit\_type WHERE id = ?";

**// Подготовка на заявката за изпълнение**

$stmt = $dbConn->prepare($query);

**// Свързване на ID-то като параметър за заявката**

$stmt->bind\_param("i", $id);

**// Изпълнение на заявката**

if ($stmt->execute()) {

**// Успешно изтриване**

echo "Успешно изтриване!";

} else {

**// Грешка при изпълнение на заявката**

echo "Грешка при изтриването.";

}

} else {

**// Грешка, ако предоставеното ID е невалидно**

echo "Невалидно ID.";

}

}

?>

* + - * + Важна функция в **Measures.php**

**function deleteUnitType(id)** {

**// Показване на диалогов прозорец за потвърждение на изтриването**

if (confirm("Сигурни ли сте, че искате да изтриете този запис?")) {

**// Изпращане на заявка до `delete\_unit\_type.php` чрез метода POST**

fetch('delete\_unit\_type.php', {

method: 'POST', **// Метод на заявката: POST**

headers: { 'Content-Type': 'application/x-www-form-urlencoded' }, **// Указваме, че изпращаме URL-кодирани данни**

body: new URLSearchParams({ id: id }).toString() **// Подготвяме тялото на заявката с ID-то като параметър**

})

.then(response => response.text()) **// Преобразуване на отговора в текст**

.then(data => {

alert(data); **// Показване на съобщение с резултата от сървъра (успех/грешка)**

location.reload(); **// Презареждане на страницата, за да се отразят промените**

})

.catch(error => {

**// Обработка на грешки, възникнали по време на заявката**

console.error('Грешка:', error**); // Логване на грешката в конзолата за отстраняване на проблеми**

alert('Грешка при изтриването.'); **// Показване на съобщение за грешка**

});

}

}

* Функционалност за автоматично изчисляване на оставащи количества съставки, базирано на използваните количества в поръчките.
* Автоматично генериране на справка за липсващи съставки, които трябва да бъдат закупени.
* Двете функционалности се случват в един файл **availability\_report.php**
* Обяснение и код на логиката ред по ред

<!DOCTYPE html>

<html>

<head>

**<!-- Определяне на кодировката на страницата като UTF-8 за поддръжка на български символи -->**

<meta charset="utf-8">

**<!-- Настройка на изгледа за мобилни устройства - така страницата ще изглежда добре на телефони и таблети -->**

<meta name="viewport" content="width=device-width, initial-scale=1">

**<!-- Включване на външен CSS файл за стилизиране на таблиците и бутоните -->**

<link rel="stylesheet" type="text/css" href="tableandbuttons.css">

**<!-- Заглавие на страницата, което ще се показва в таба на браузъра -->**

<title =>Справка за съставките</title>

</head>

<body>

<?php

**// Включване на навигационното меню**

include "manu.php";

**// Включване на конфигурационния файл за връзка с базата данни**

include "config.php";

**// Зареждаме всички съставки от таблицата 'ingredients'**

$query = "SELECT id, name, quantity FROM ingredients";

$result = mysqli\_query($dbConn, $query);

$ingredients = []; **// Масив за съхраняване на съставките**

**// Обхождаме резултатите от заявката и ги добавяме в масива $ingredients**

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

$ingredients[$row['id']] = [

'name' => $row['name'], **// Името на съставката**

'quantity' => $row['quantity'], **// Количеството на съставката**

];

}

**// Сега зареждаме информация за съставките, използвани в поръчките**

$query = "

SELECT

i.id AS ingredient\_id, **// ID на съставката**

i.name, **// Име на съставката**

i.quantity AS total\_quantity, **// Общо количество на съставката**

SUM(od.quantity \* di.quantity) AS total\_used **// Изчисляваме общото количество на съставката, използвано в поръчките**

FROM

ingredients i

JOIN

dish\_ingredients di ON i.id = di.ingredient\_id **// Свързваме съставката с таблицата dish\_ingredients**

JOIN

order\_dishes od ON di.dish\_id = od.dish\_id **// Свързваме ястията с поръчките**

GROUP BY

i.id **// Групираме по ID на съставката**

";

$result = mysqli\_query($dbConn, $query);

**// Масив, в който ще съхраняваме информацията за необходимите съставки**

$needed\_ingredients = [];

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

$ingredientId = $row['ingredient\_id']; **// Получаваме ID на съставката**

**// Добавяме информацията за съставката в масива**

$needed\_ingredients[$ingredientId] = [

'total\_used' => $row['total\_used'], **// Използвано количество**

'remaining' => isset($ingredients[$ingredientId]) ? $ingredients[$ingredientId]['quantity'] - $row['total\_used'] : 0, **// Оставащо количество (разлика между наличното и използваното)**

];

}

**// Извеждаме резултатите в HTML таблица**

echo "<h2 style=\"text-align: center;\">Справка за съставките</h2>"; **// Заглавие на таблицата**

echo "<table border='1'>"; **// Създаваме таблица с рамка**

echo "<tr><th>Съставка</th><th>Необходими количества</th><th>Оставащи количества</th><th>Необходими за покупка</th></tr>"; **// Заглавия на колоните**

**// Обхождаме масива с необходимите съставки и ги извеждаме ред по ред в таблицата**

foreach ($needed\_ingredients as $ingredientId => $data) {

$ingredientName = $ingredients[$ingredientId]['name']; **// Вземаме името на съставката**

$totalUsed = $data['total\_used']; **// Вземаме общото количество използвано**

$remaining = $data['remaining']; **// Вземаме оставащото количество**

$needed = $remaining < 0 ? abs($remaining) : 0; **// Ако остават отрицателни количества, означава, че трябва да се закупи още (превръщаме в положително число)**

**// Извеждаме ред в таблицата с информацията за съставката**

echo "<tr>

<td>$ingredientName</td> **<!-- Името на съставката -->**

<td>$totalUsed</td> **<!-- Използваното количество -->**

<td>" . ($remaining >= 0 ? $remaining : 0) . "</td**> <!-- Оставащото количество (показва само положителните стойности) -->**

<td>$needed</td> **<!-- Количеството, което трябва да се закупи (показва 0, ако няма нужда от покупка) -->**

</tr>";

}

**// Затваряме таблицата**

echo "</table>";

**// Затваряме връзката с базата данни**

mysqli\_close($dbConn);

?>

</body>

</html>

* + 1. **Логика за зависими изтривания:** При изтриване на ястие или поръчка, всички свързани записи в междинните таблици също се изтриват автоматично.
* Обяснение на код в **delete\_dish.php** ред по ред. **delete\_ingredient.php** и **delete\_order.php** работят по подобен начин. **delete\_dish.php** показва логиката и за двата случая – невъзможност за изтриване, ако има поръчка и изтриване на свързващите колони в таблицата, за това този файл е най – подходящ за пример и обяснение.

<?php

**// Включване на конфигурационния файл, който съдържа информация за връзката с базата данни**

include "config.php";

**// Проверяваме дали заявката е POST (т.е. дали формата е подадена)**

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

**// Проверка дали е подаден параметър 'id' чрез POST и дали той е число**

if (isset($\_POST['id']) && is\_numeric($\_POST['id'])) {

**// Ако ID е предоставено и е число, го превръщаме в цяло число**

$dishId = intval($\_POST['id']);

**// Проверка дали dishId е валидно (не е 0 или отрицателно)**

if ($dishId) {

**// Подготвяме заявка, за да проверим дали ястието е свързано с поръчки**

$checkQuery = "SELECT COUNT(\*) as count FROM order\_dishes WHERE dish\_id = ?";

$stmt = $dbConn->prepare($checkQuery); **// Подготвяме заявката за изпълнение**

$stmt->bind\_param("i", $dishId); **// Свързваме параметъра с променливата $dishId**

$stmt->execute(); **// Изпълняваме заявката**

$result = $stmt->get\_result()->fetch\_assoc(); **// Получаваме резултата от заявката**

$stmt->close(); **// Затваряме изричната подготовка на заявката**

**// Ако резултатът показва, че ястието е част от поръчка, не можем да го изтрием**

if ($result['count'] > 0) {

echo "Ястието не може да бъде изтрито, защото е поръчано."; **// Извеждаме съобщение**

} else {

**// Ако ястието не е свързано с поръчки, подготвяме заявка за изтриване на ястието**

$query = "DELETE FROM dishes WHERE id = ?";

$stmt = $dbConn->prepare($query); **// Подготвяме заявката за изпълнение**

$stmt->bind\_param("i", $dishId); **// Свързваме параметъра с променливата $dishId**

**// Изпълняваме заявката и проверяваме дали е успешна**

if ($stmt->execute()) {

echo "Ястието и свързаните съставки бяха успешно изтрити."; **// Извеждаме съобщение за успешното изтриване**

} else {

echo "Грешка при изтриване на ястието: " **. $stmt->error; // Извеждаме съобщение за грешка**

}

$stmt->close(); **// Затваряме изричната подготовка на заявката**

}

} else {

echo "Невалидно ID."; **// Ако ID не е валидно, извеждаме съобщение за грешка**

}

}

}

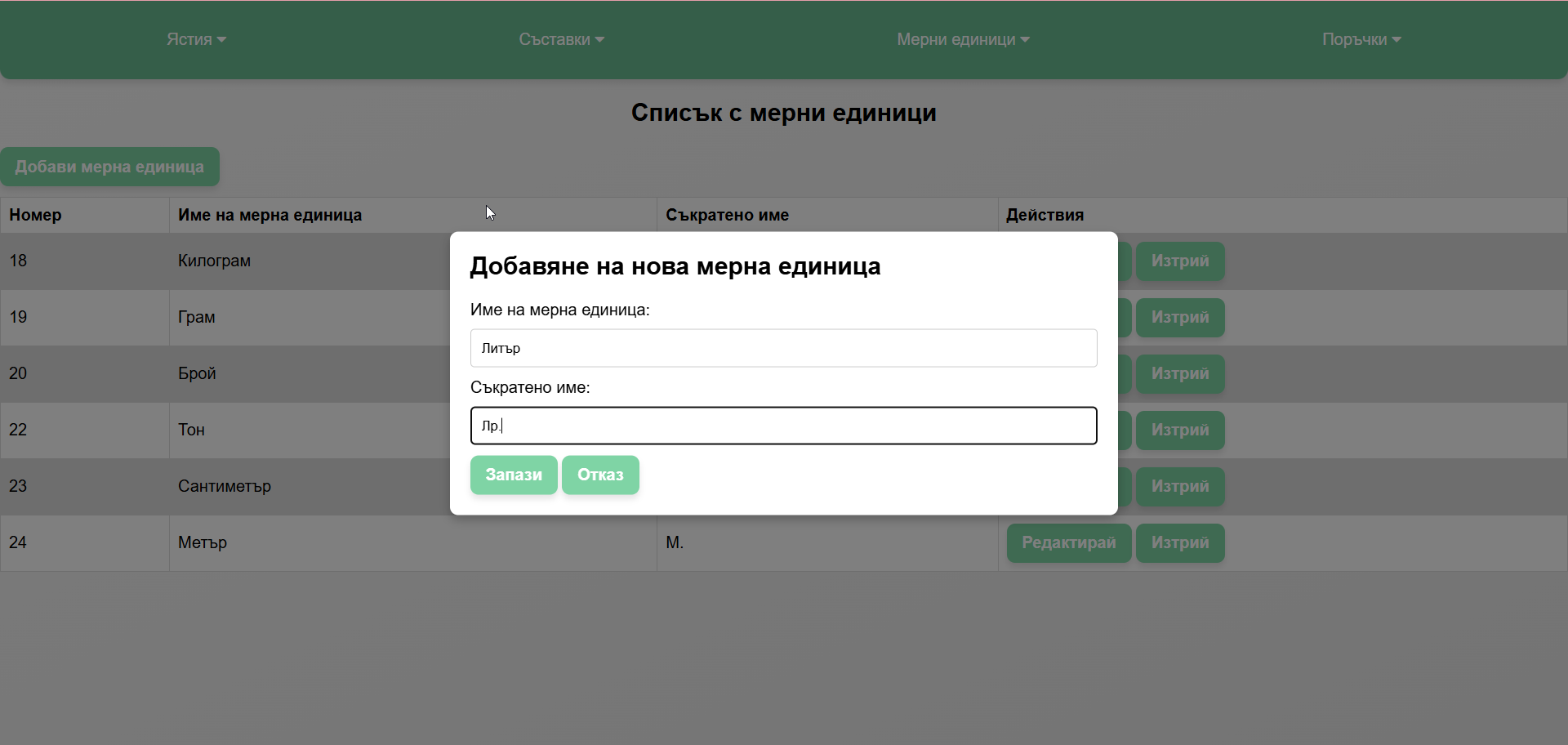
**// Затваряме връзката с базата данни**

$dbConn->close();

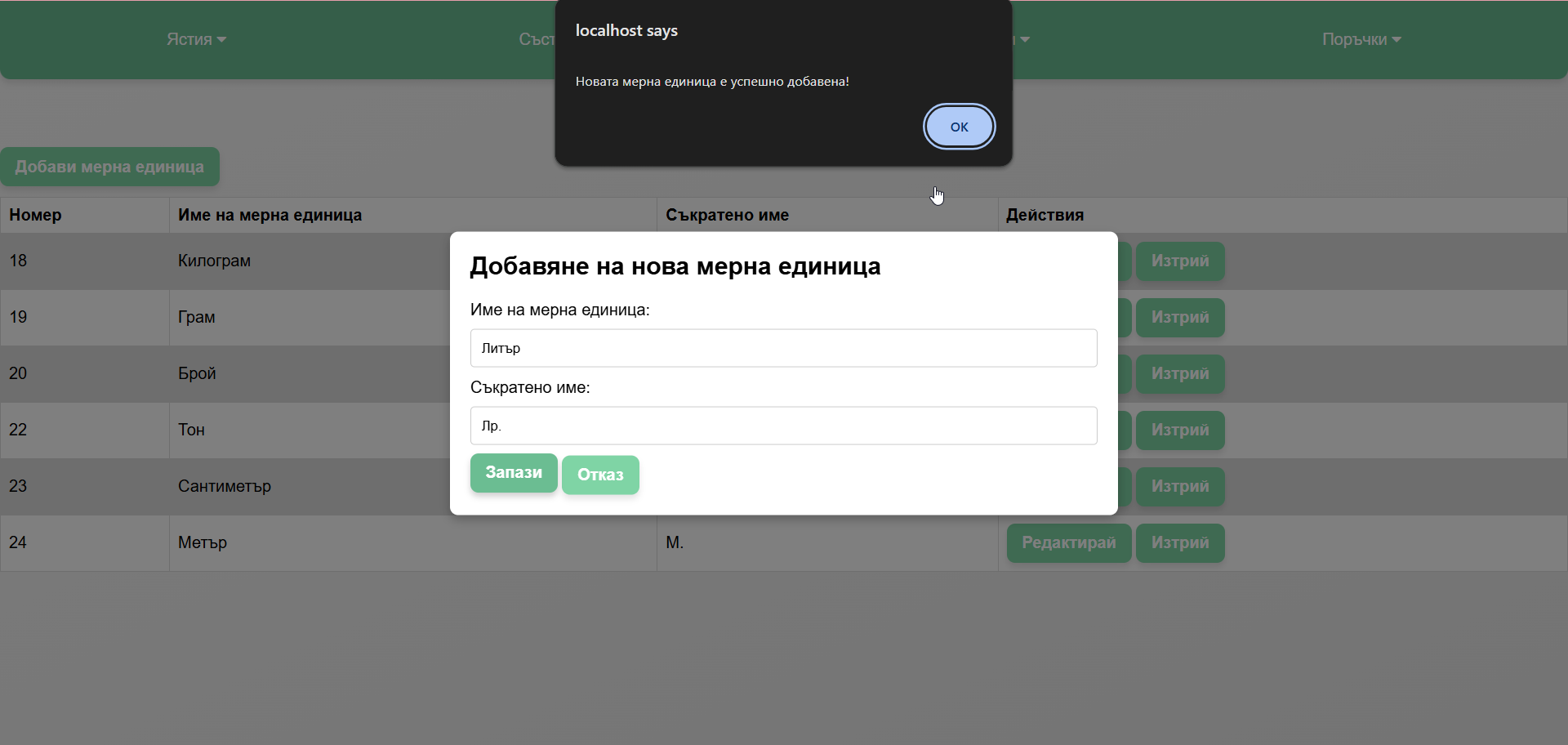
?>

1. **Кратко ръководство за потребителя. Тестване на приложение. Изводи и възможности за развитие.**
   1. **Кратко ръководство за потребителя**
      1. Приложението съдържа меню за по – лесна употреба.
      2. От навигационната лента може да се избира между четири таба.
      3. Когато се сложи мишката на надписа „Ястия“, „Съставки“, „Мерни единици“ или „Поръчки“ се отваря падащо меню. Направено е по такъв начин, за да може да се разшири по – лесно при добавяне на нови функционалности.

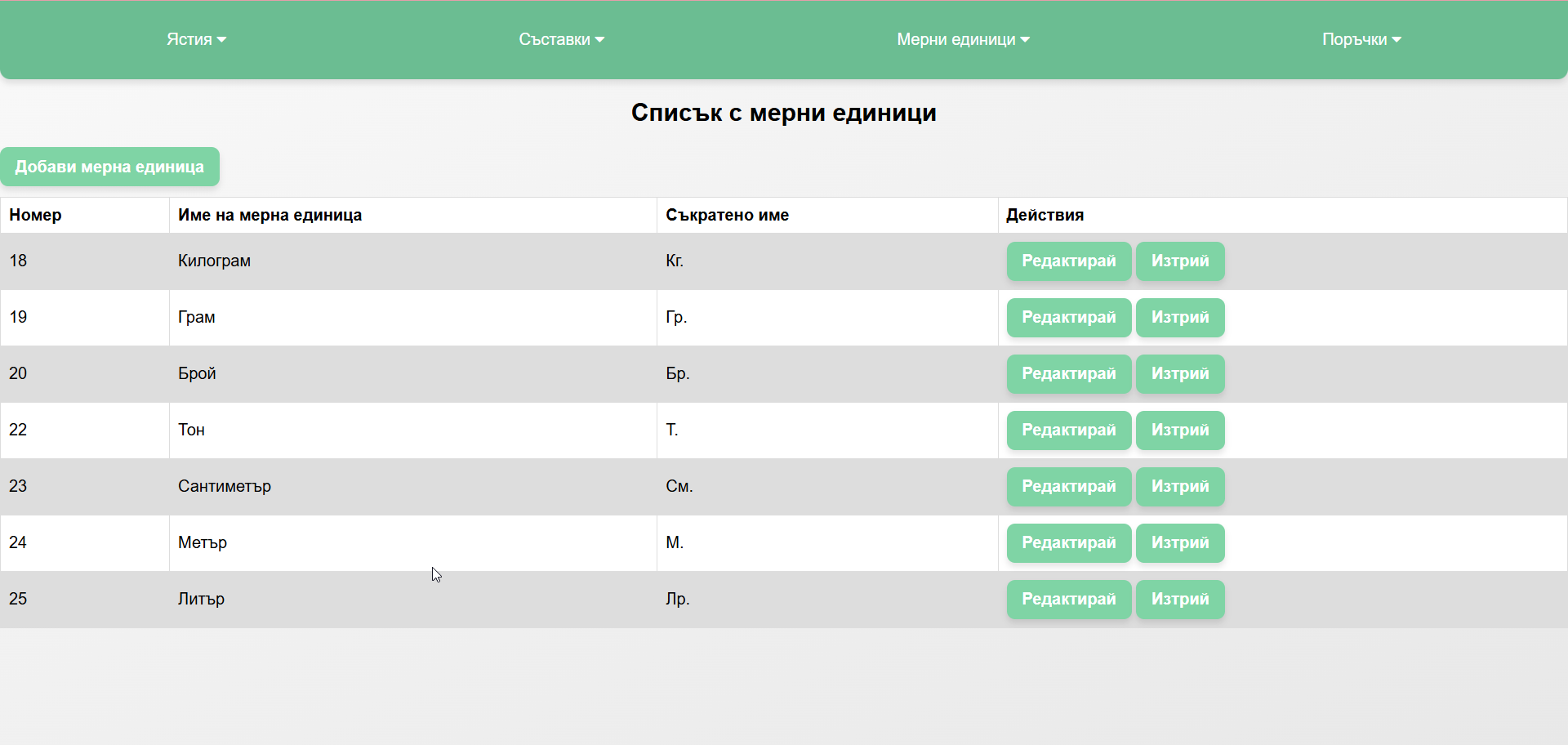
* „Ястия“ :
* Таблица с информация за всички ястия.
* Бутон за добавяне, който при натискане отваря „Поп ъп“ прозорец да въвеждане на ново ястие.
* Бутон за редакция, който при натисне отваря „Поп ъп“ прозорец за редакция избраното ястие.
* Бутон за изтриване, който при натисне след съгласие изтрива избраното ястие.
* „Съставки“ състой се от избор „Съставки“ и „Отчет на съставки“ :
* „Съставки“:
* Таблица с информация за всички съставки.
* Бутон за добавяне, който при натискане отваря „Поп ъп“ прозорец да въвеждане на нова съставка.
* Бутон за редакция, който при натисне отваря „Поп ъп“ прозорец за редакция избраната съставка.
* Бутон за изтриване, който при натисне след съгласие изтрива избраната съставка.
* „Отчет на съставки“:
* Таблица с информация за използваните, наличните и нужните за покупка съставки.
* „Мерни единици“:
* Таблица с информация за всички мерни единици.
* Бутон за добавяне, който при натискане отваря „Поп ъп“ прозорец да въвеждане на нова мерна единица.
* Бутон за редакция, който при натисне отваря „Поп ъп“ прозорец за редакция избраната мерна единица.
* Бутон за изтриване, който при натисне след съгласие изтрива избраната мерна единица.
* „Поръчки“ :
* Таблица с информация за всички поръчки.
* Бутон за добавяне, който при натискане отваря „Поп ъп“ прозорец да въвеждане на нова поръчка.
* Бутон за редакция, който при натисне отваря „Поп ъп“ прозорец за редакция избраната поръчка.
* Бутон за изтриване, който при натисне след съгласие изтрива избраната поръчка.
  1. **Тестване на приложението**
     1. **Мерни единици**
* Добавяне:
* Добавяне на информация: „Литър“ , „Лр.“



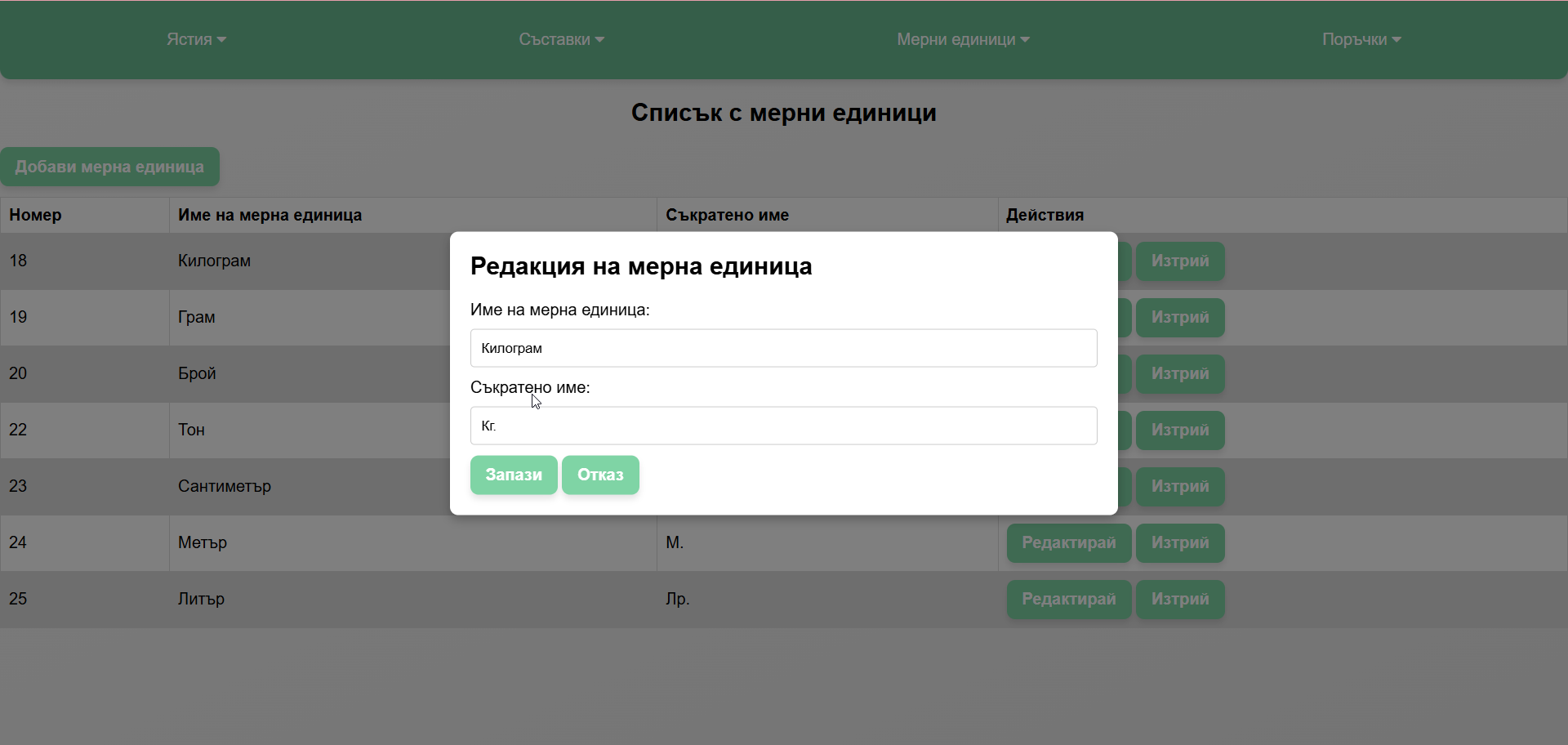
* Съобщение към потребителя при успех или при грешка



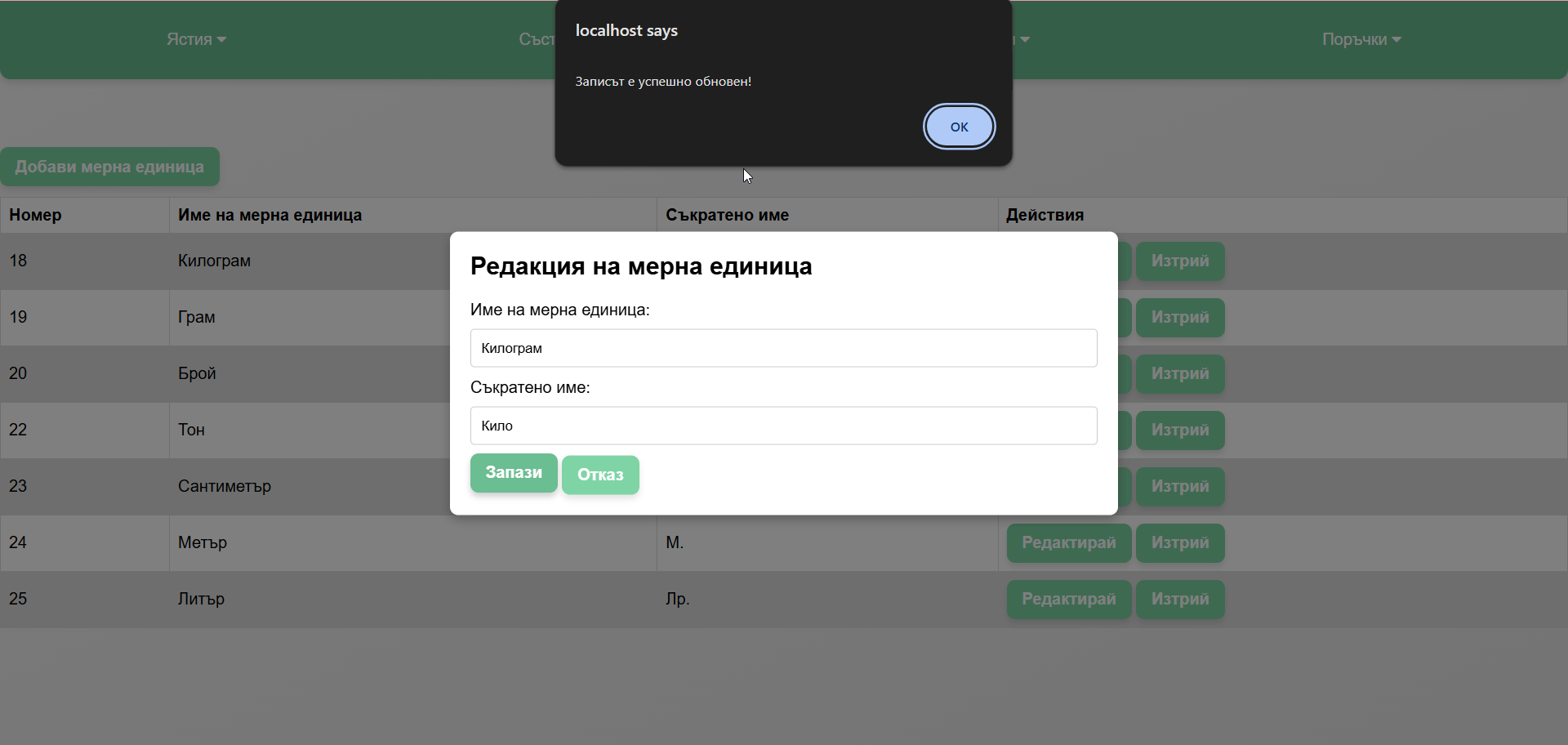
* Автоматично обновяване на таблицата след изпълнение



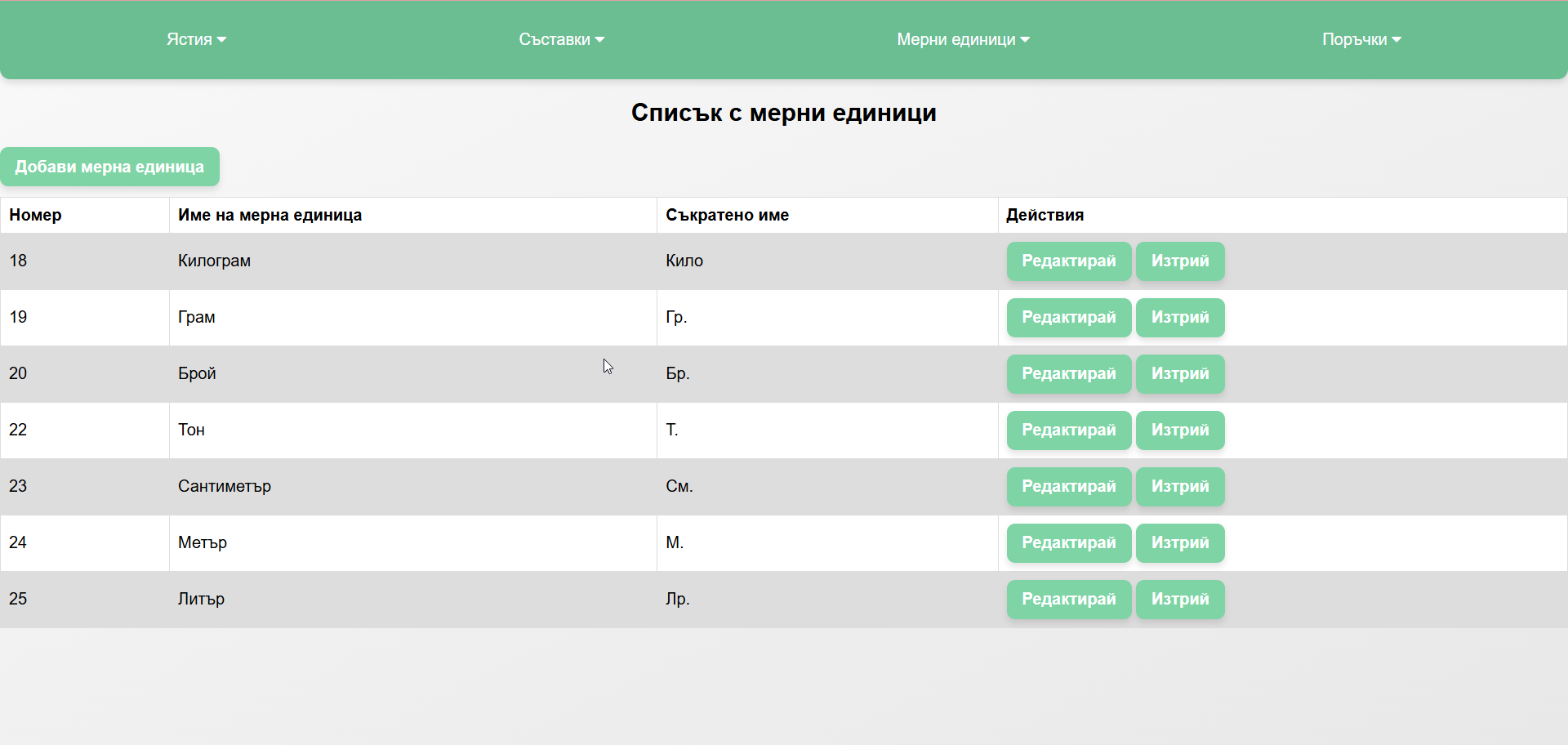
* Редакция:
* При избор на ред от таблицата



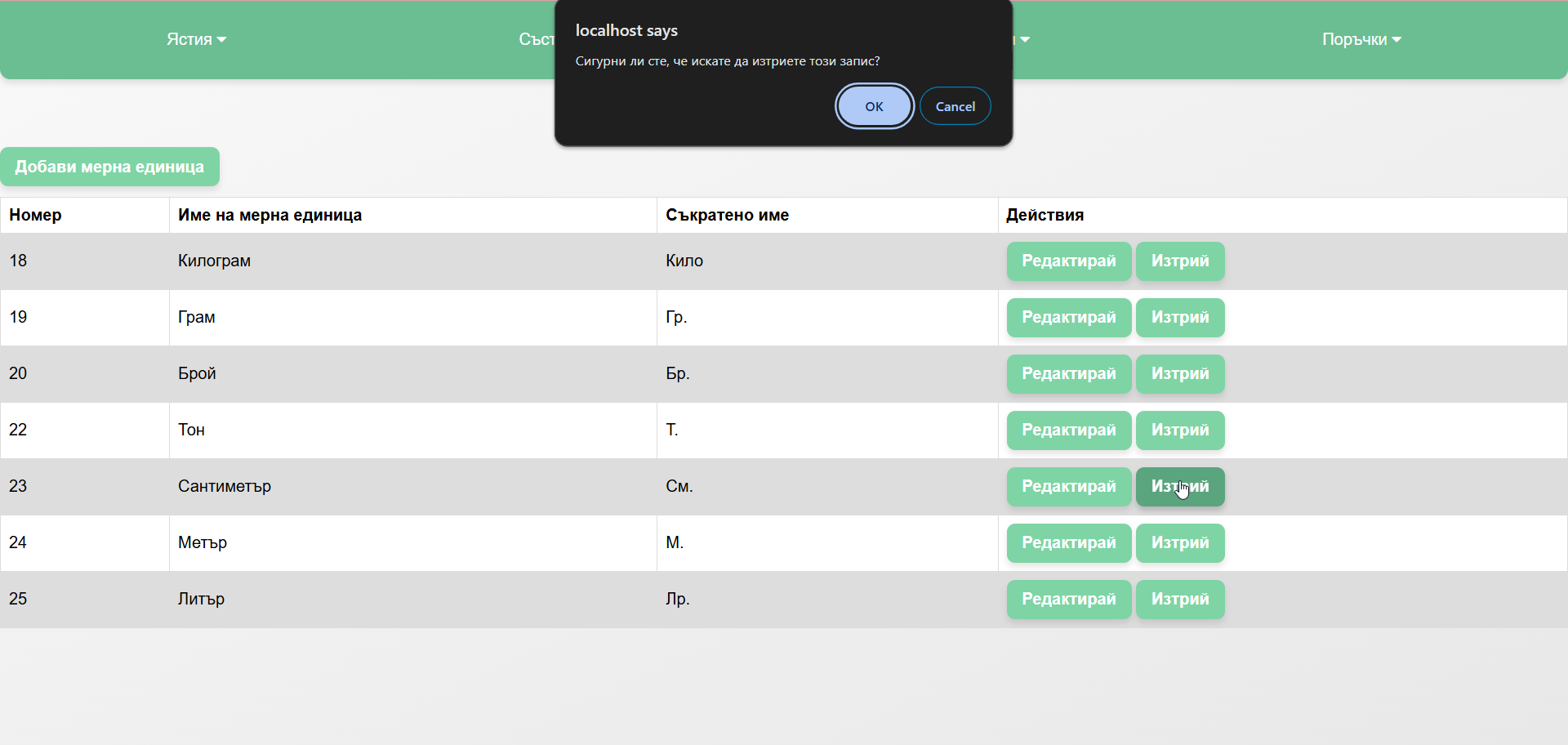
* Съобщение за успех или грешка



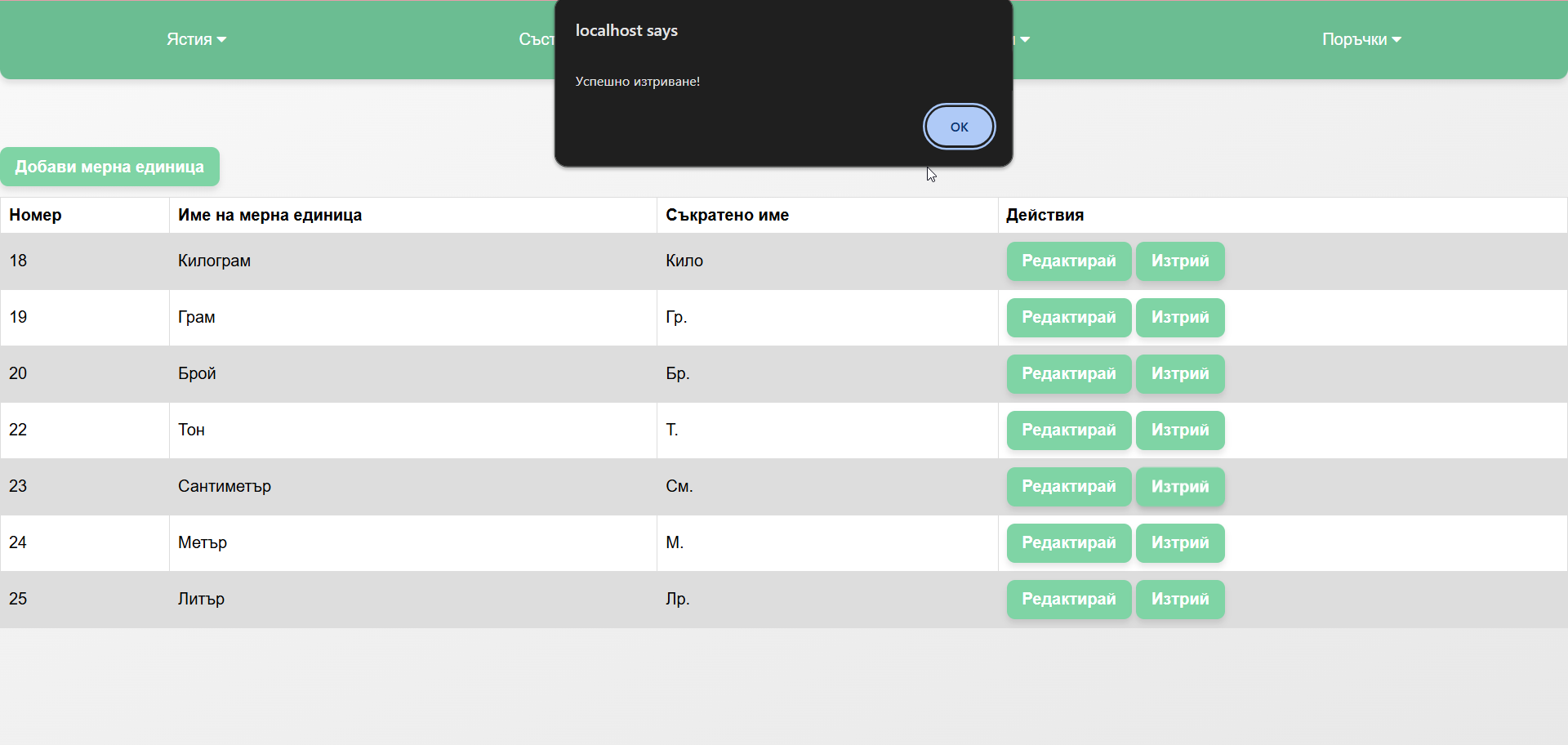
* Автоматично обновяване на таблицата след промяна



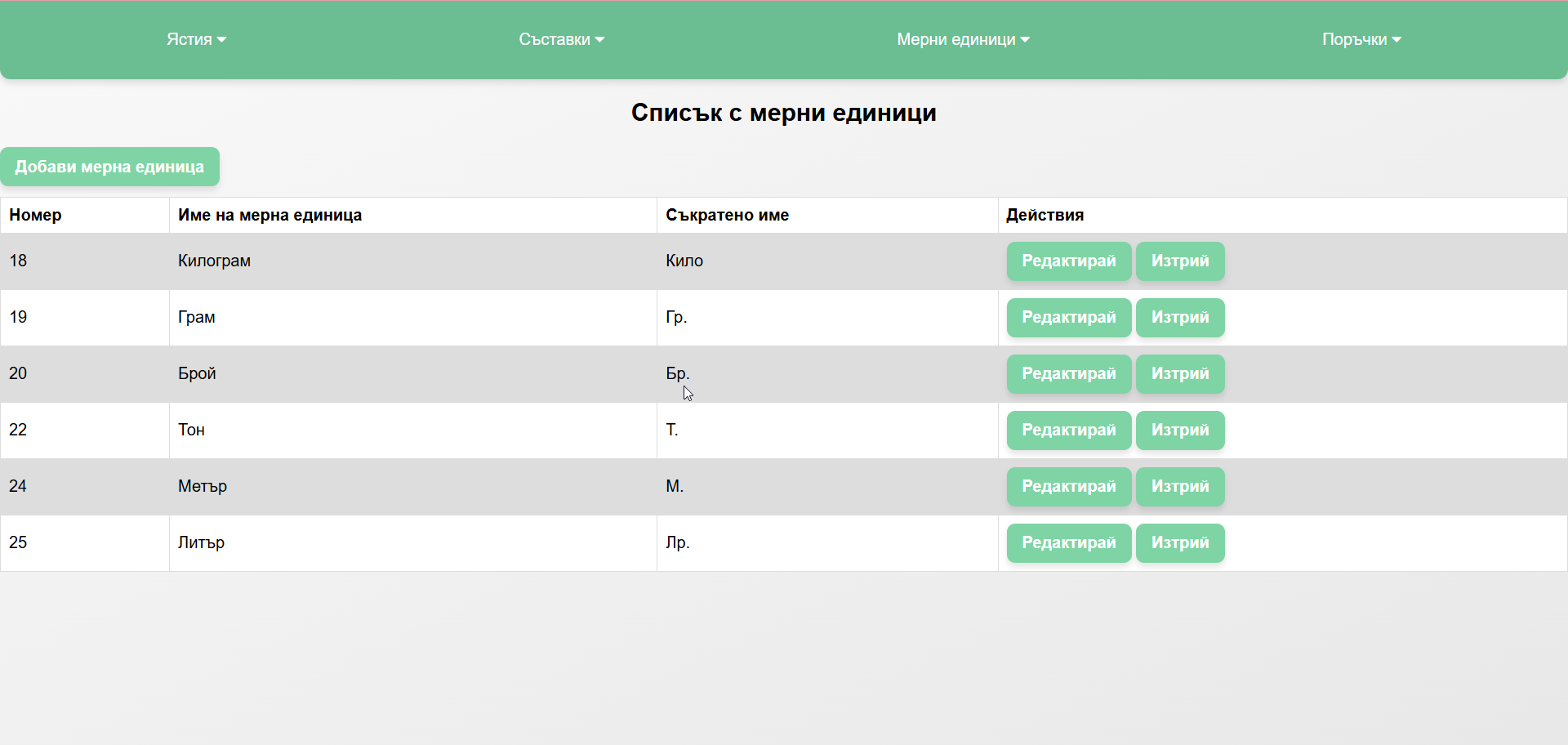
* Изтриване:
* Съобщение за съгласието на потребителя при избор на ред от таблицата



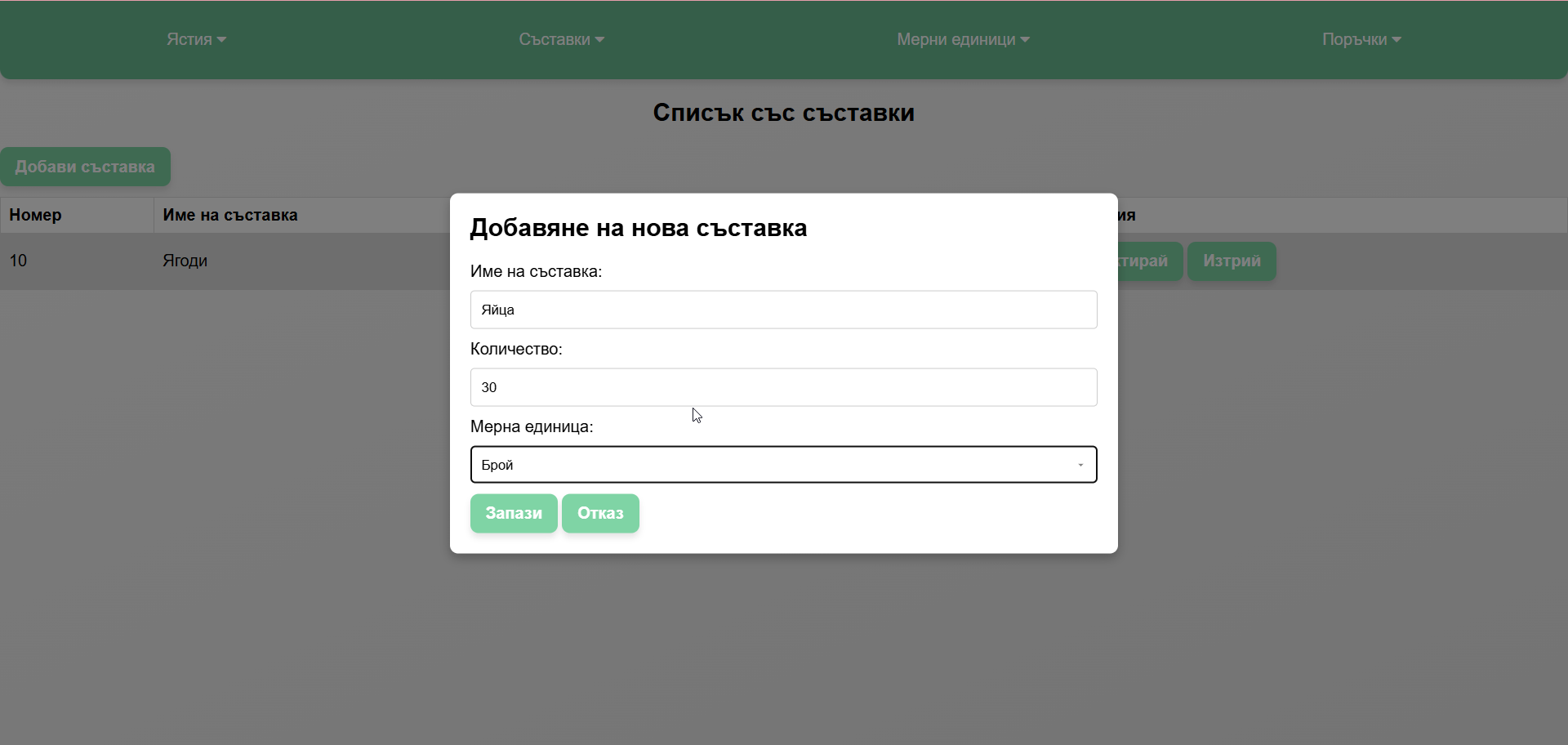
* Съобщение за успех или грешка



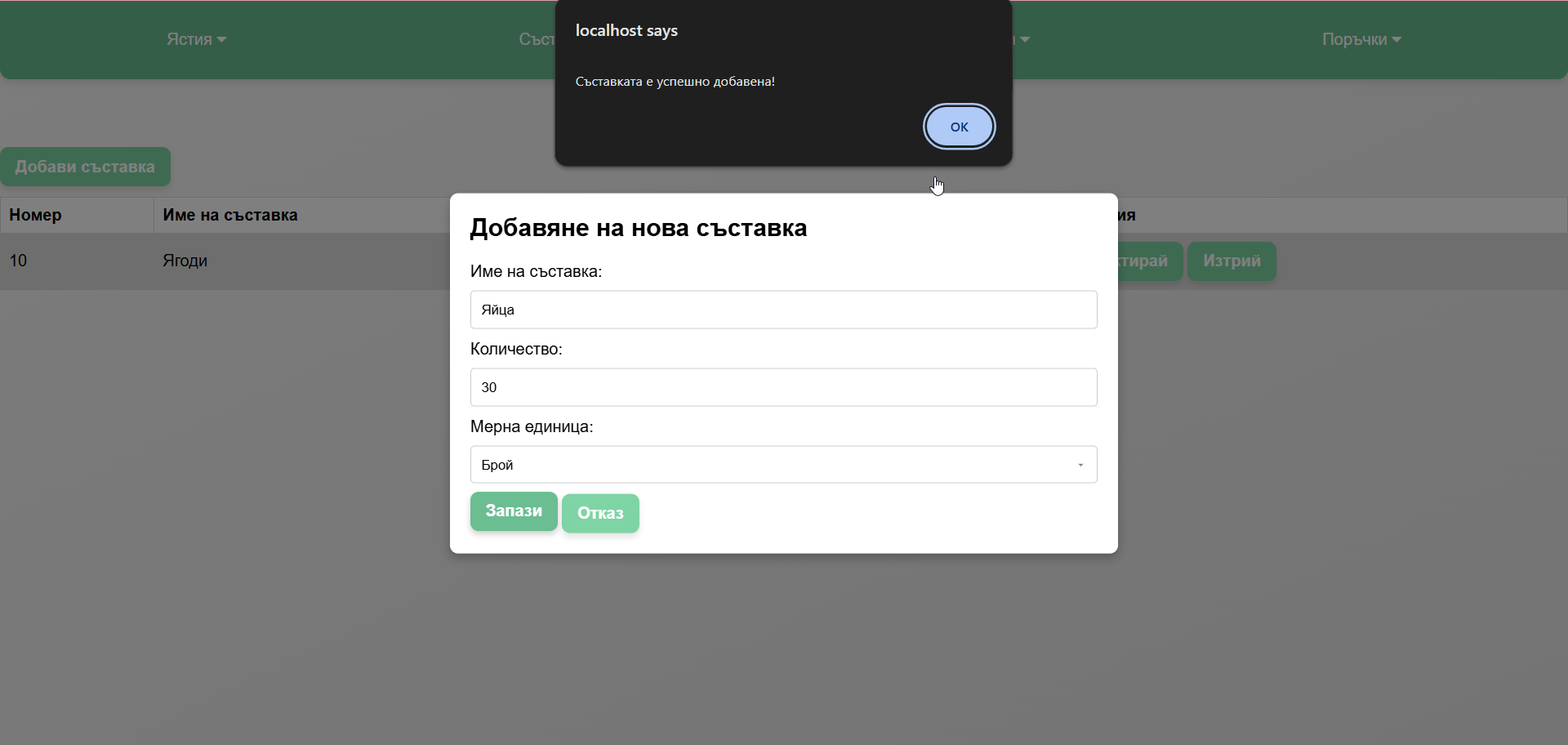
* Автоматично обновяване на таблицата



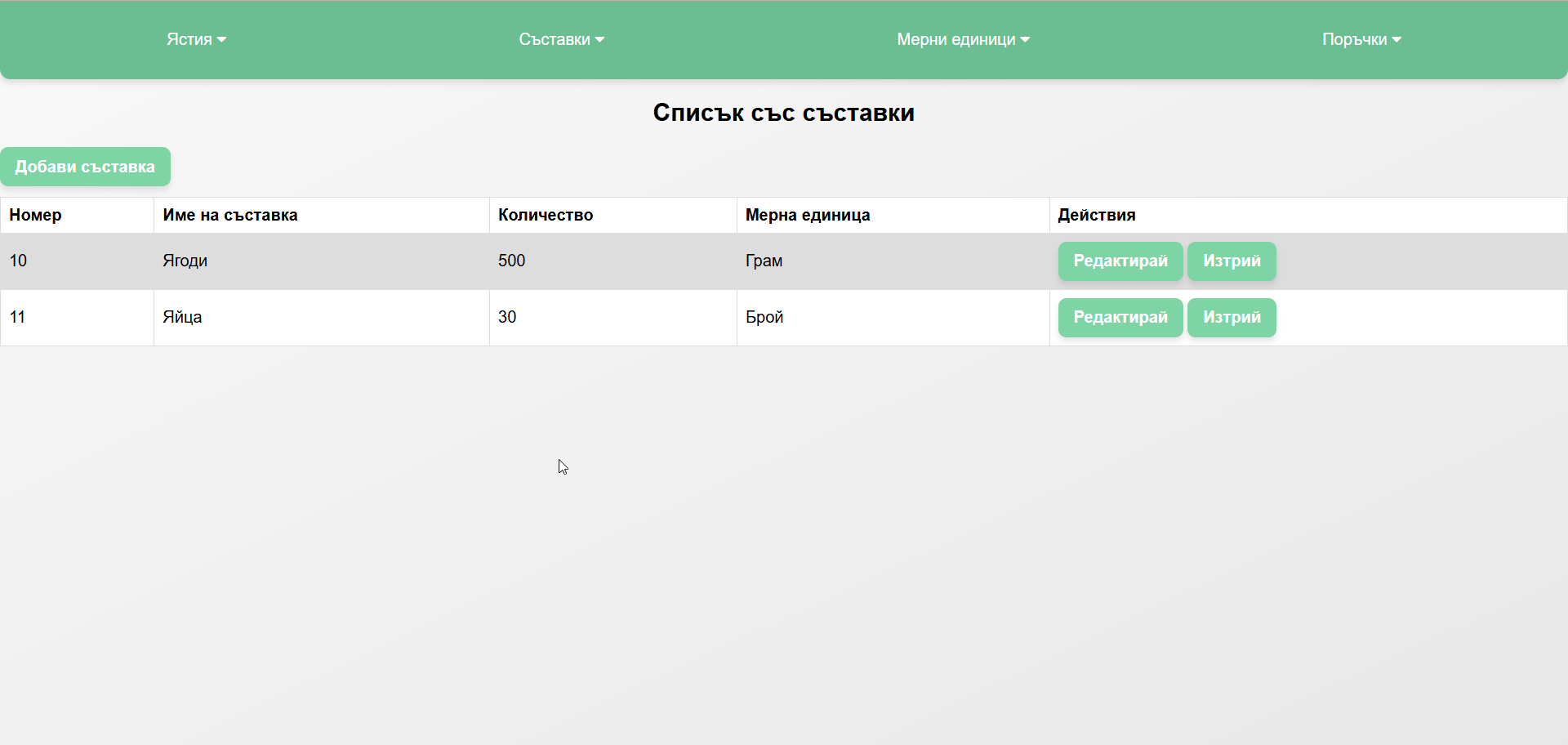
* + 1. **Съставки**
* Добавяне:
* Въвеждаме информация „Яйца“, 30 и избираме от падащото меню, че са в брой



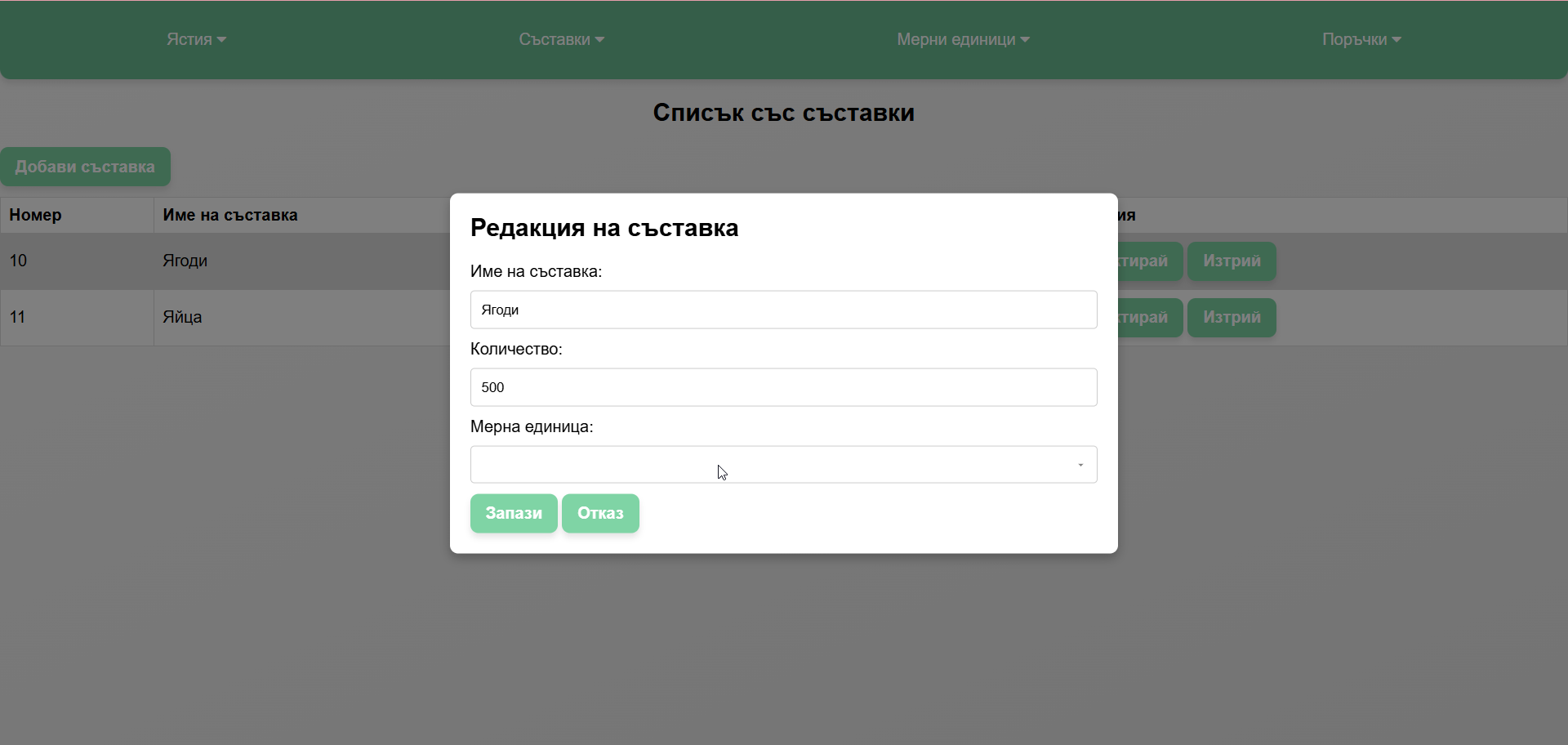
* Съобщение за успех или грешка



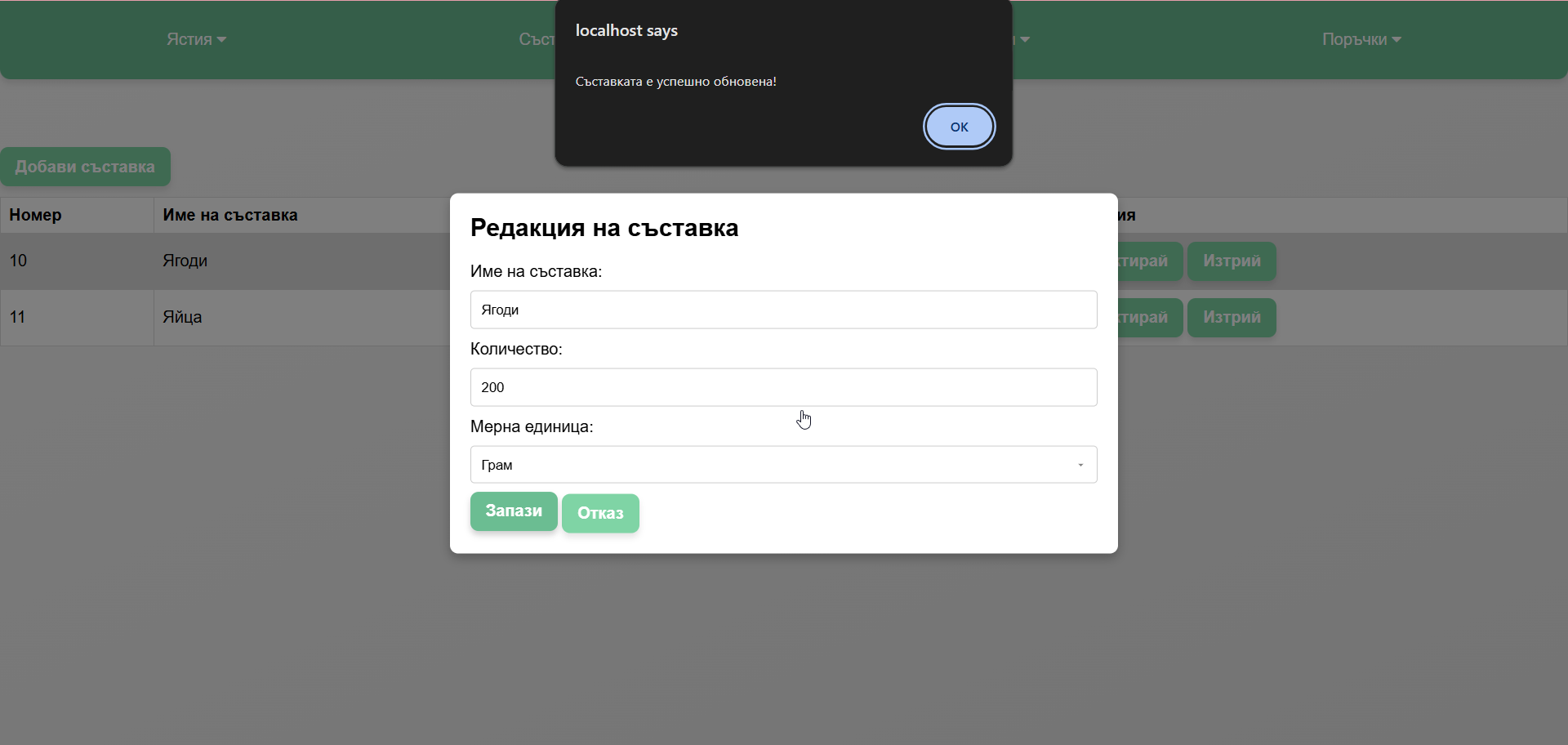
* Автоматично обновяване след добавяне



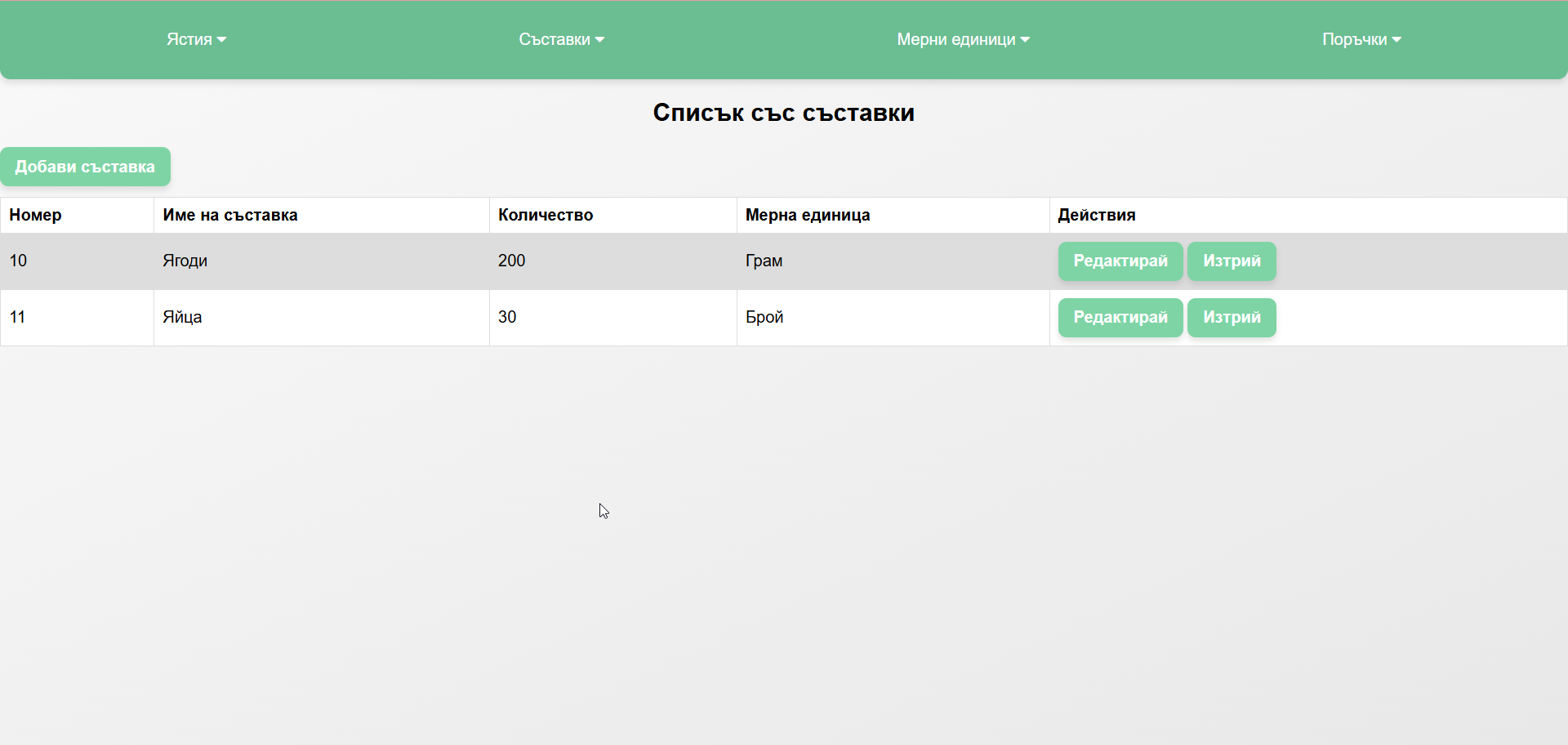
* Редакция:
* След натискане на ред от таблица. Тук може да се подобри, тъй като не зарежда в падащото меню, като избрана стойност, информацията за мерната единица.



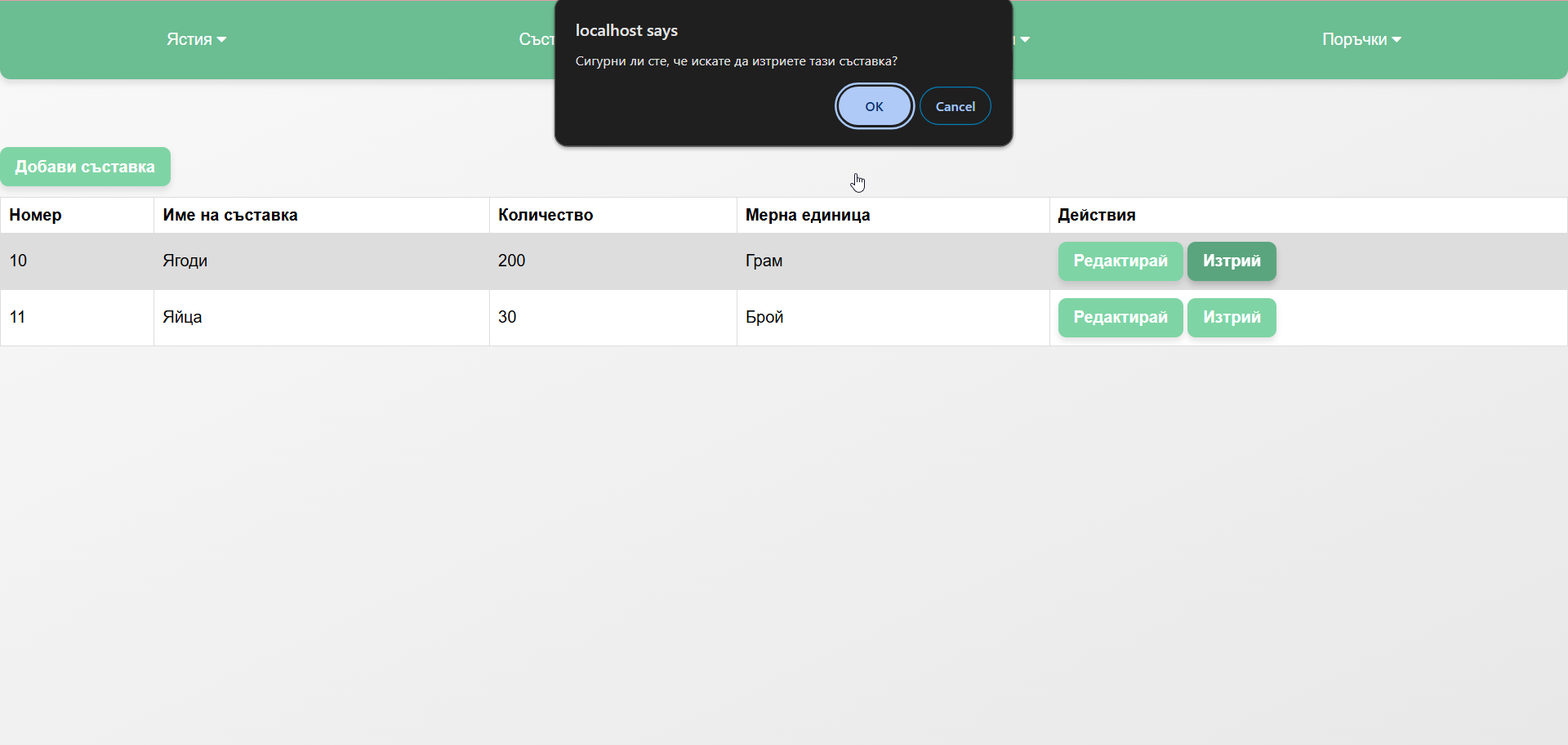
* Съобщение за грешка или успех



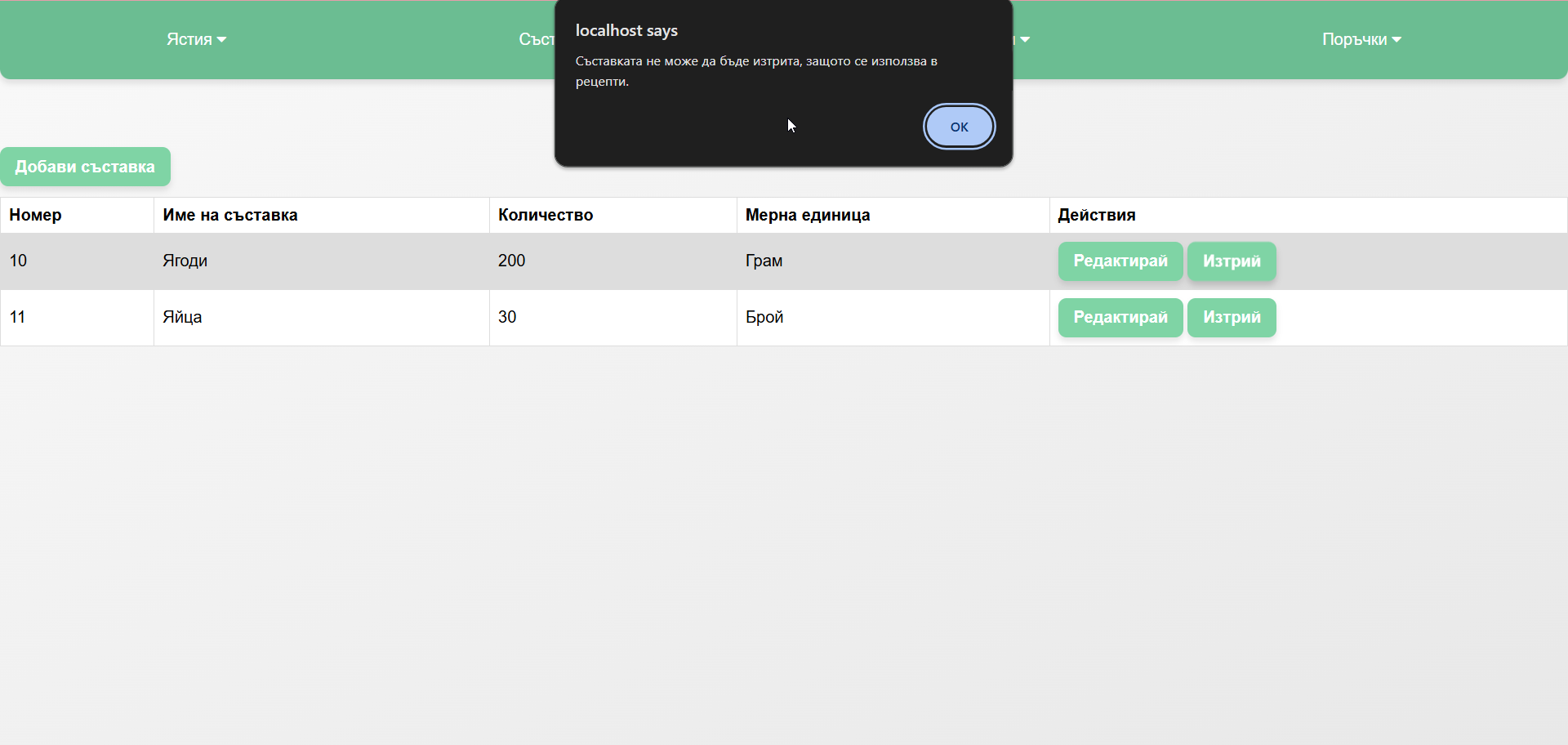
* Автоматично обновяване на таблицата след редакция



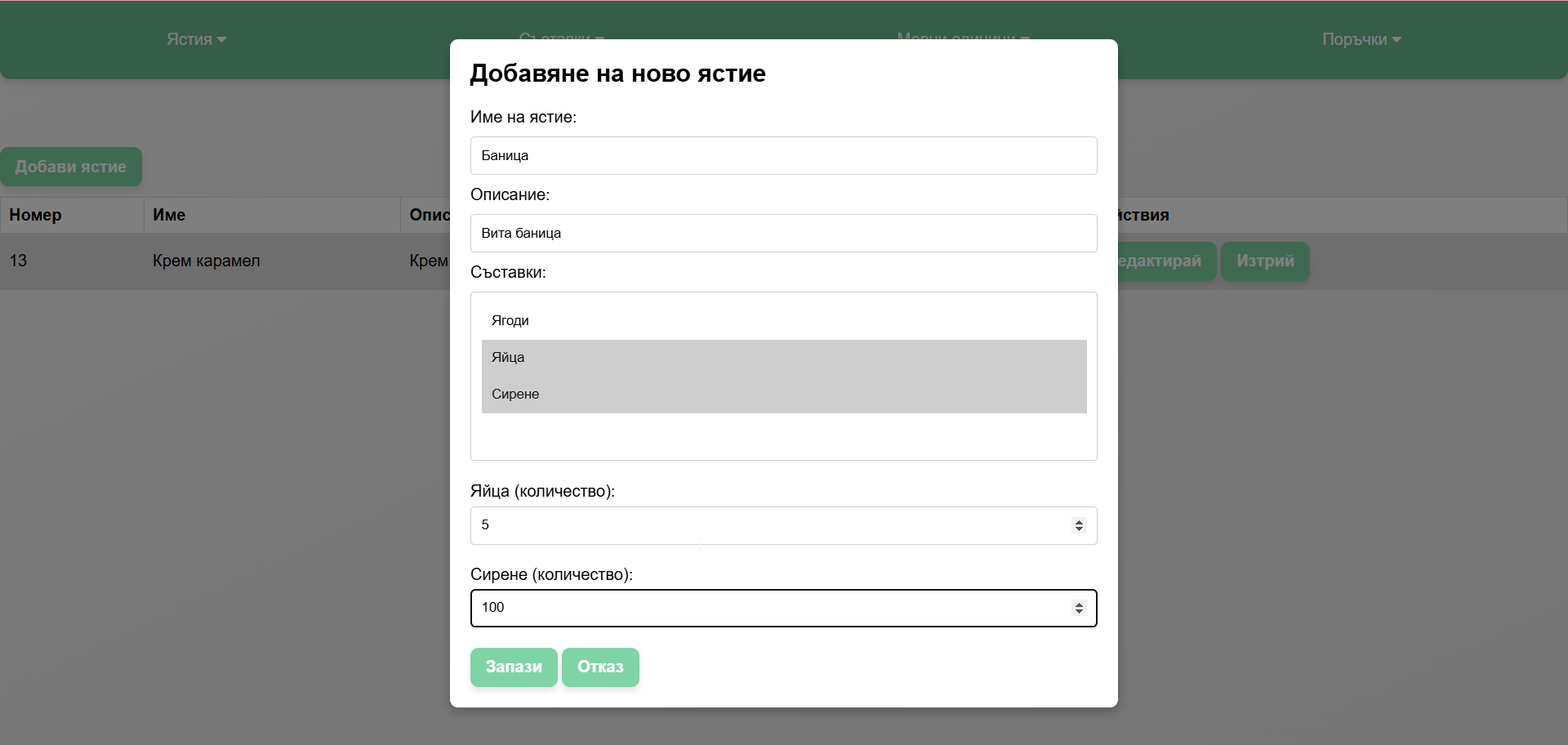
* Изтриване:
* Изискване на съгласие за изтриване при избор на ред от таблица



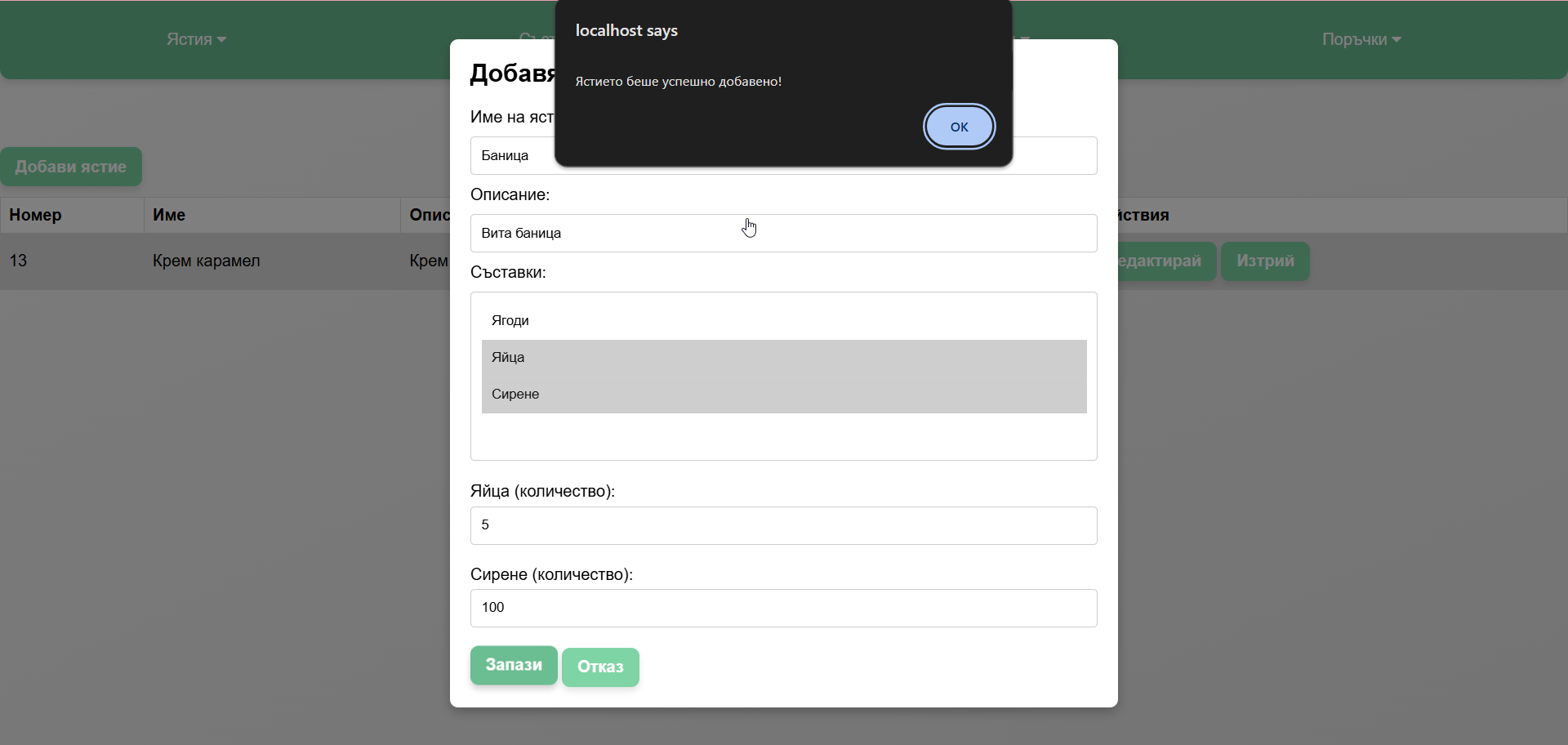
* Съобщение, ако се използва тази съставка в ястие



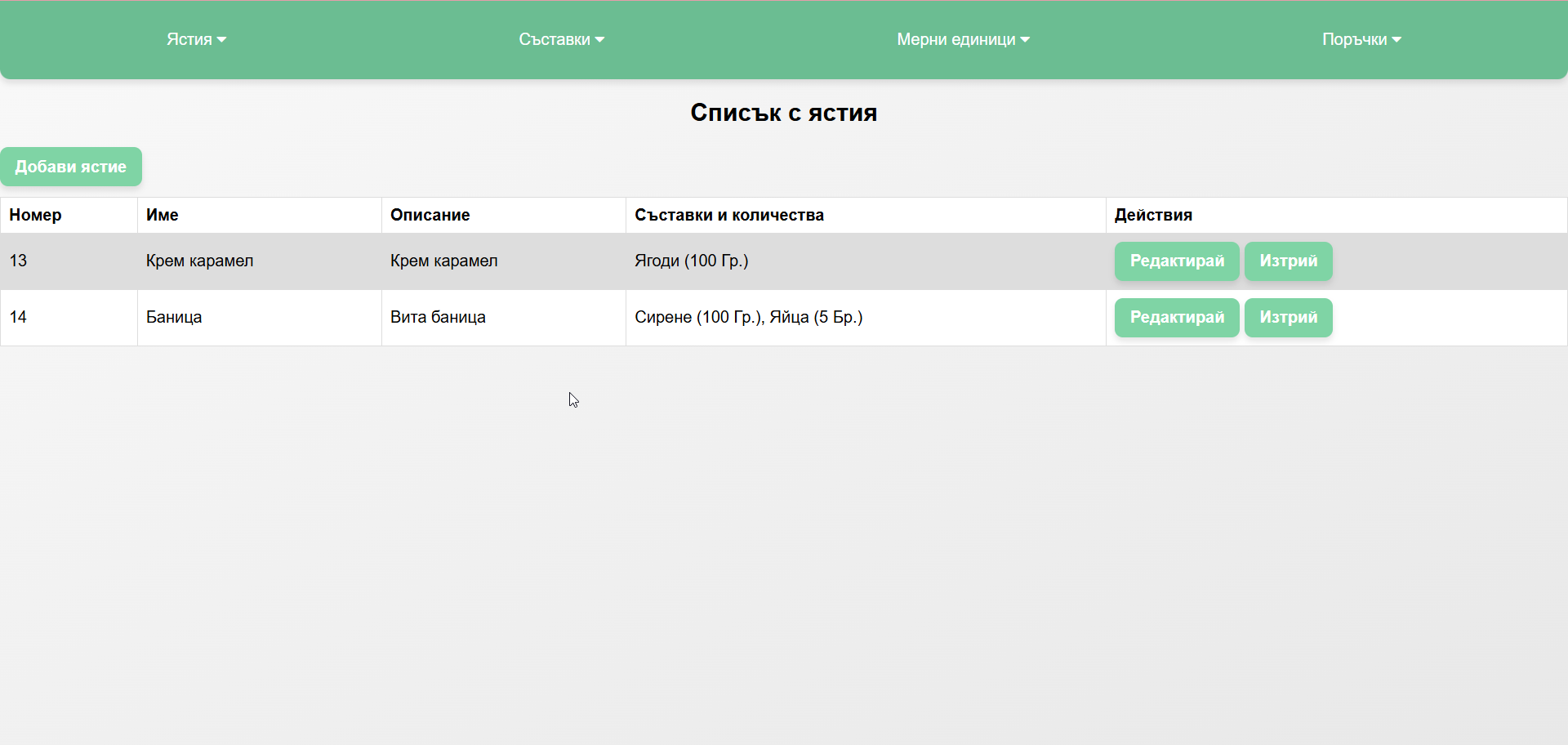
* + 1. **Ястия**
* Добавяне
* Добавяне на информация за „Баница“ в описание „Вита баница“ и съставки „Яйца - 5“ и „Сирене 100“



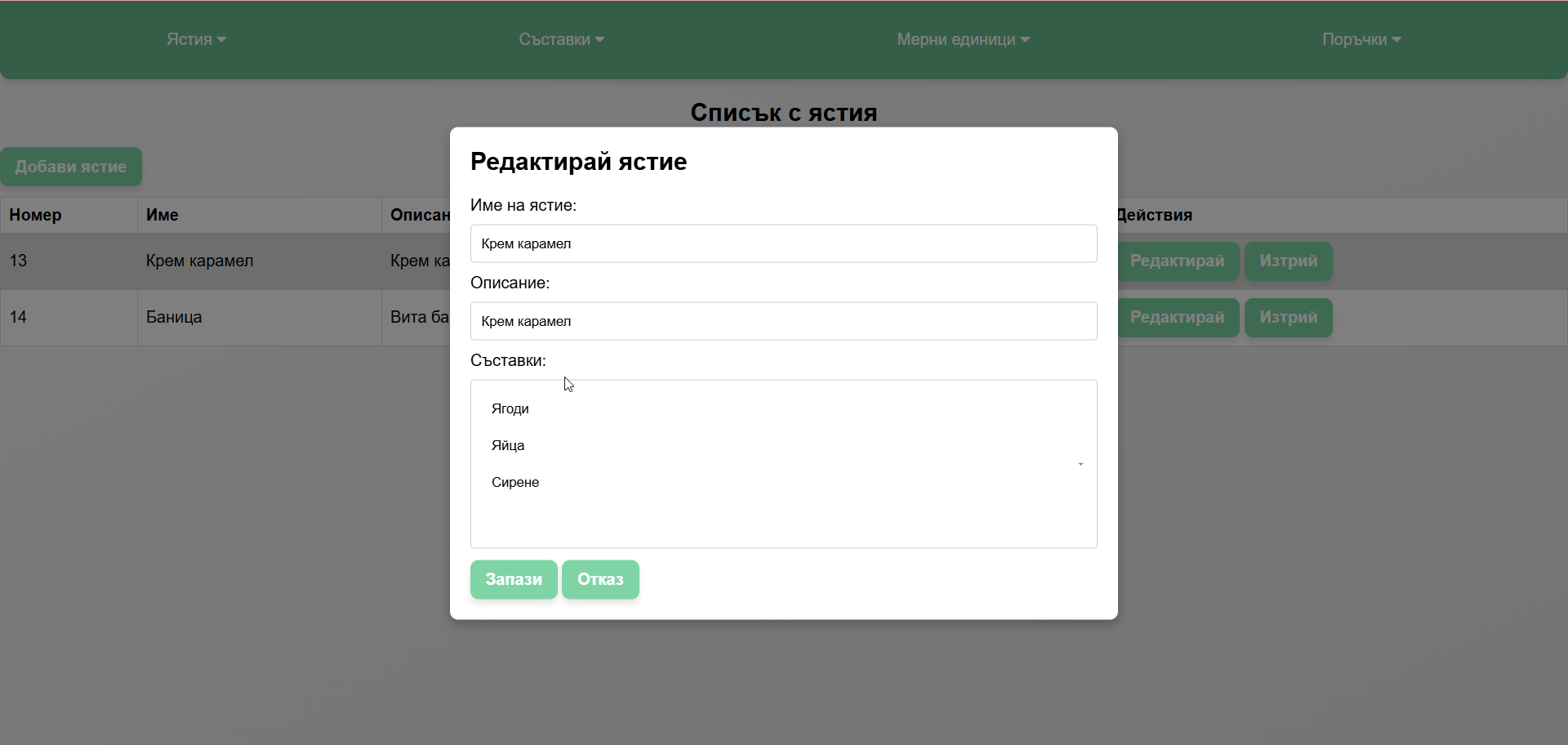
* Съобщение за успех или грешка



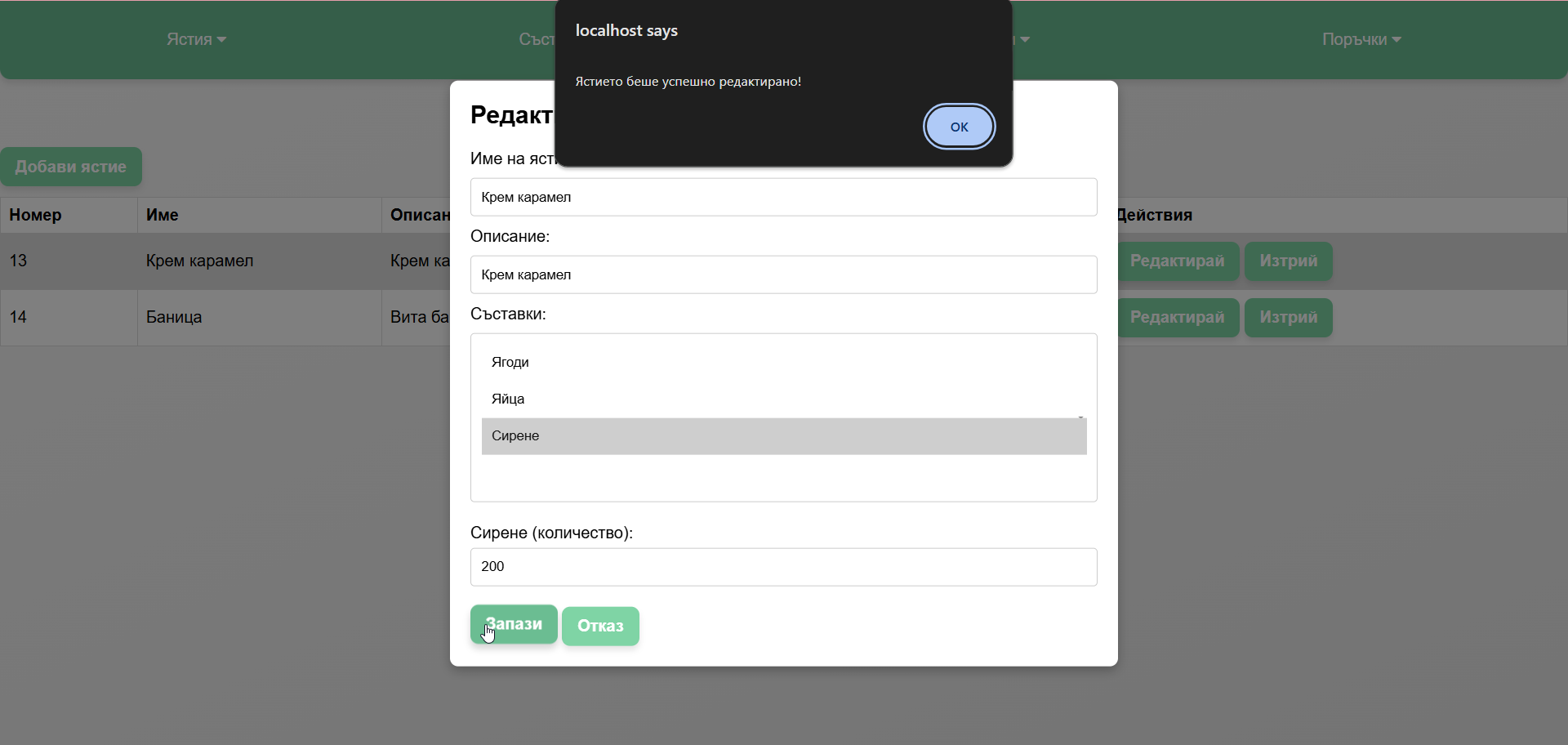
* Автоматично обновяване на таблицата след добавяне



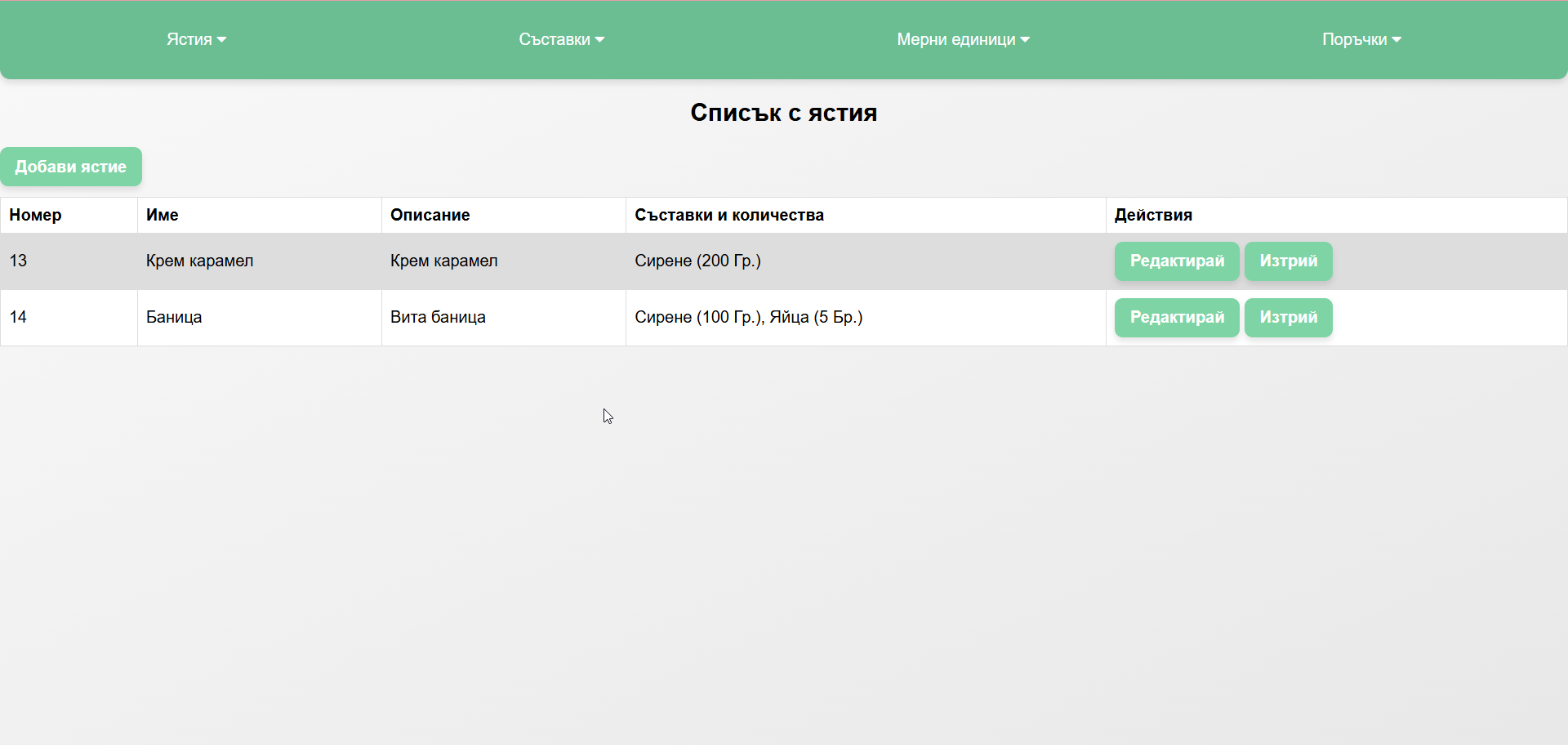
* Редакция
* При избор на ред от таблицата. Тук може да се подобри, тъй като не зарежда вече избраните продукти.



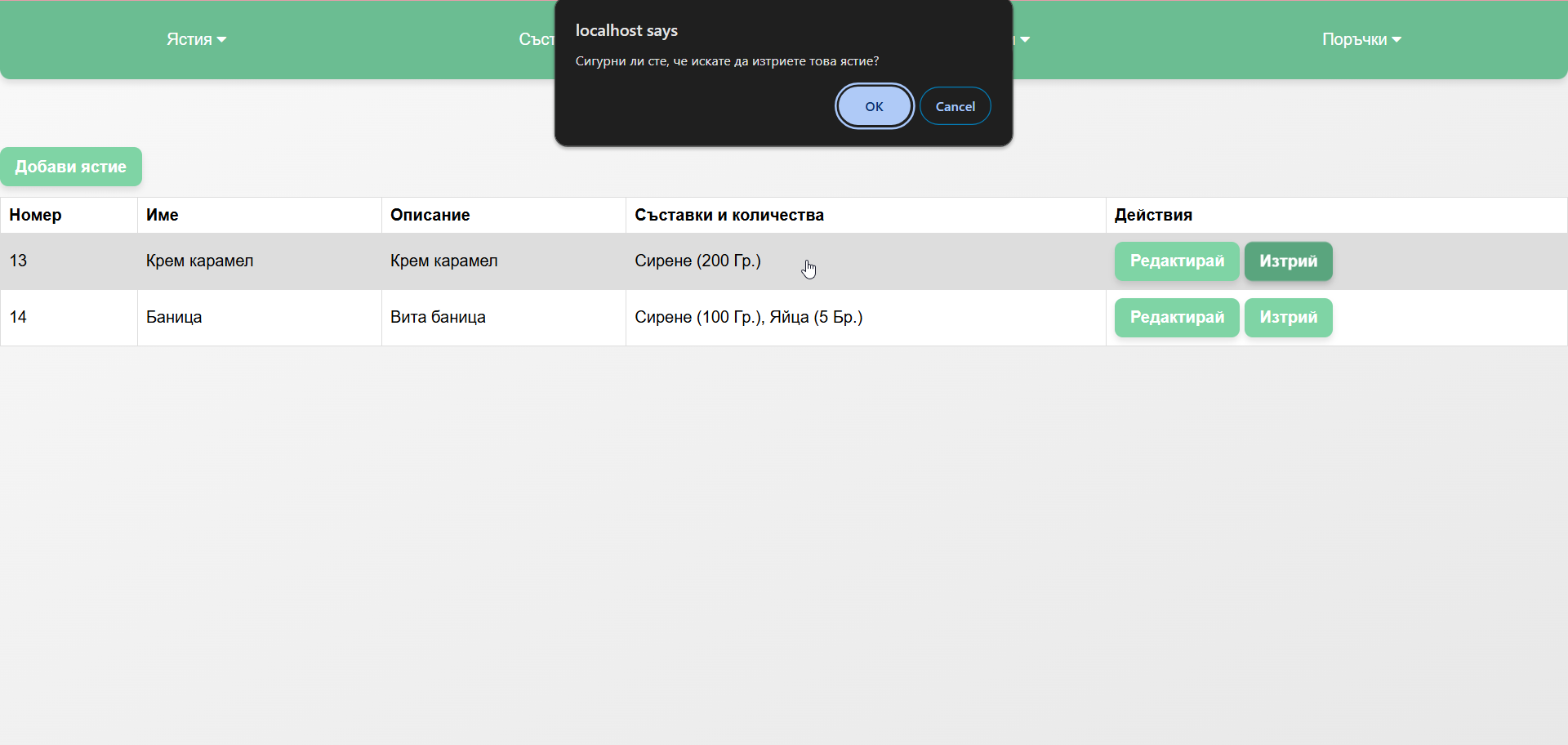
* Съобщение за успех или грешка



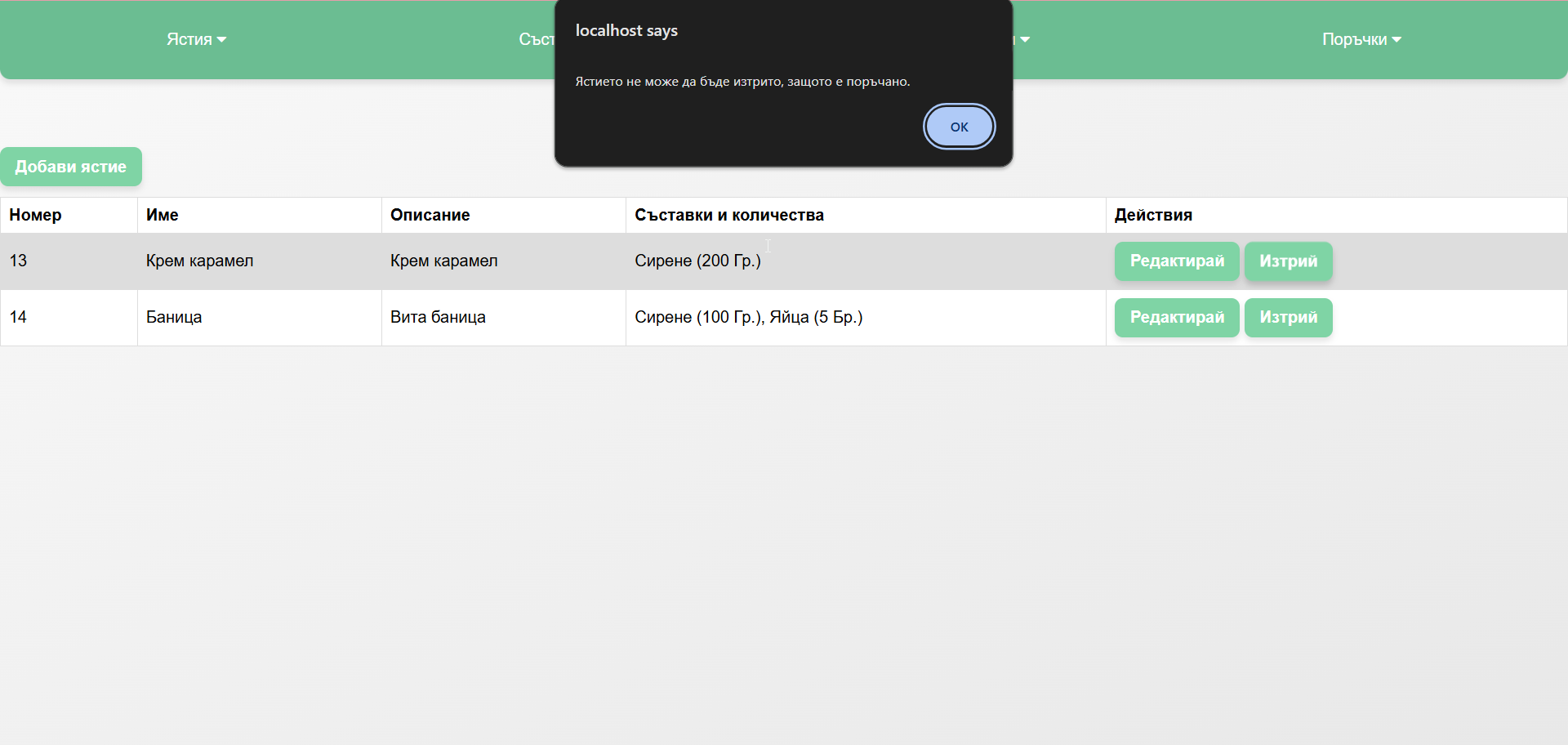
* Автоматично обновяване след редакция



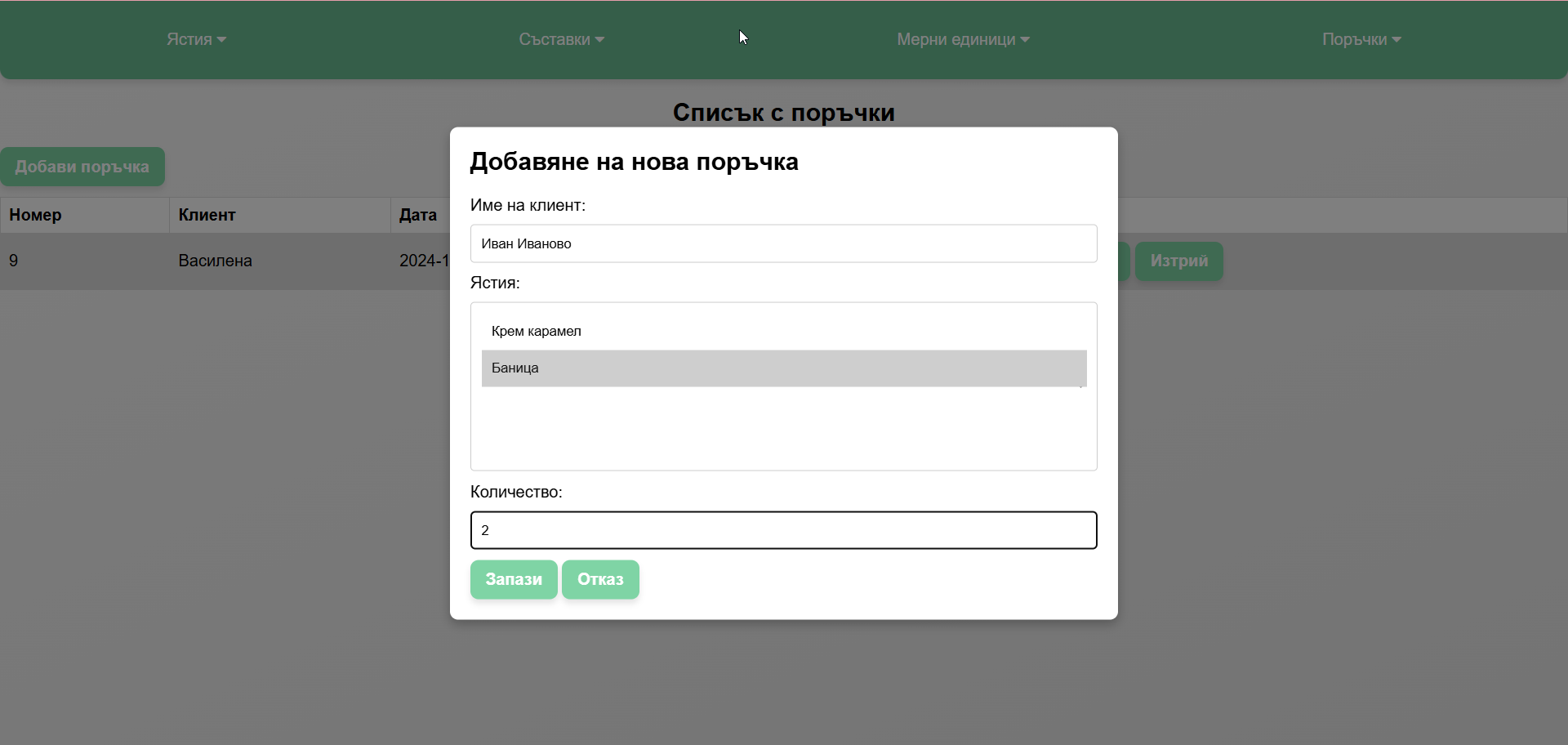
* Изтриване
* Съобщение за съгласие при избор на ред от таблицата



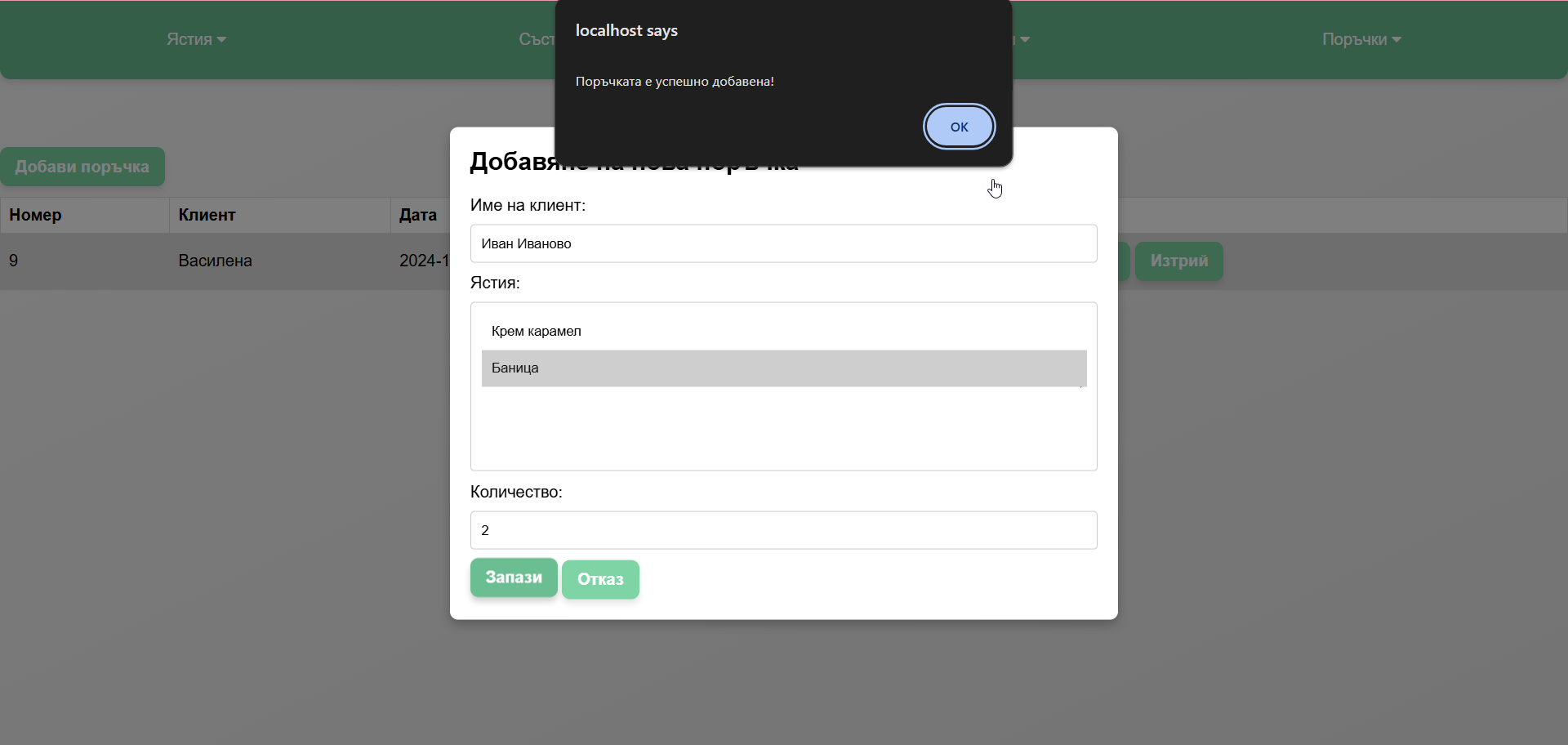
* Съобщение, ако има поръчано такова ястие



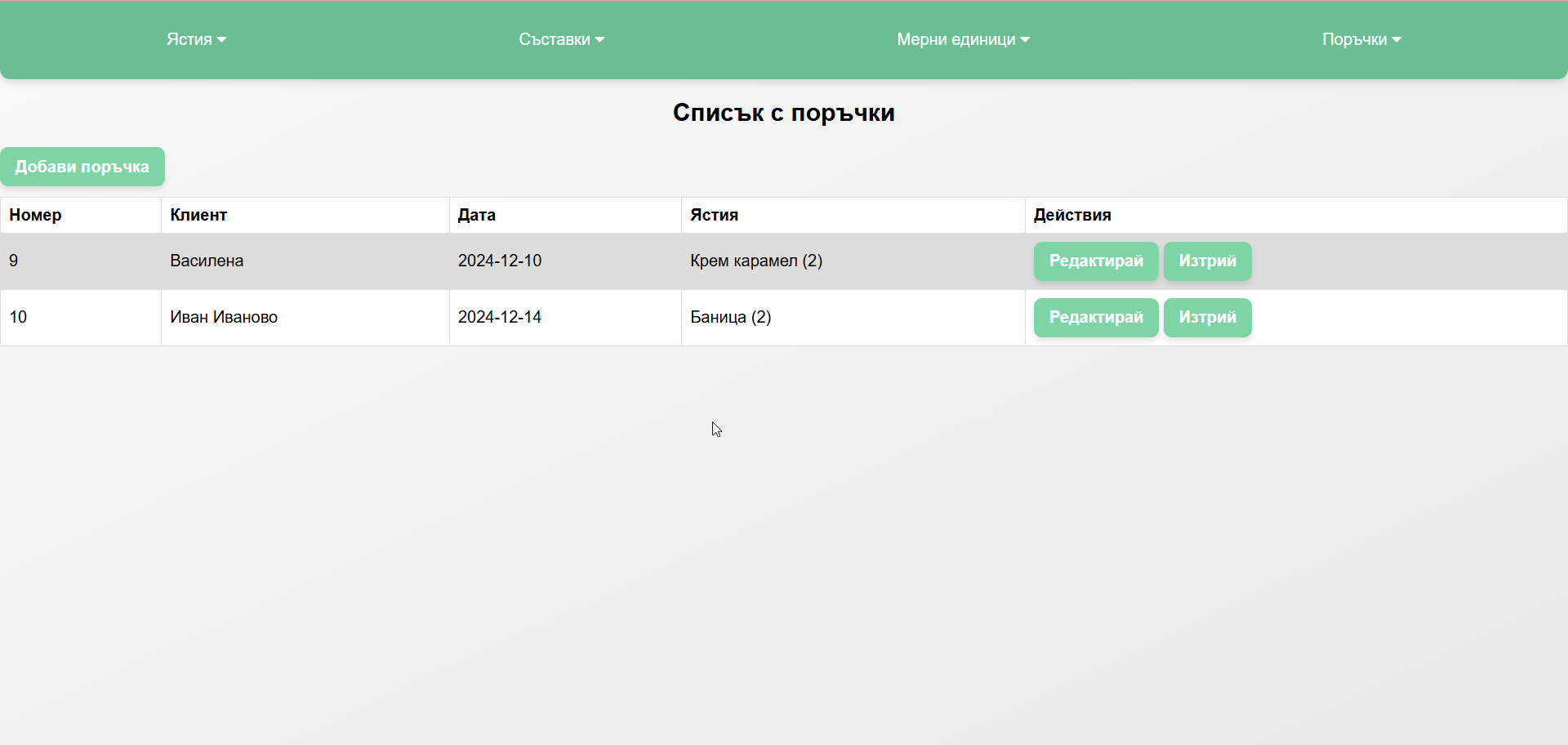
* + 1. **Поръчки**
* Добавяне
* Добавяне на поръка за Иван Иванов с 2 баници



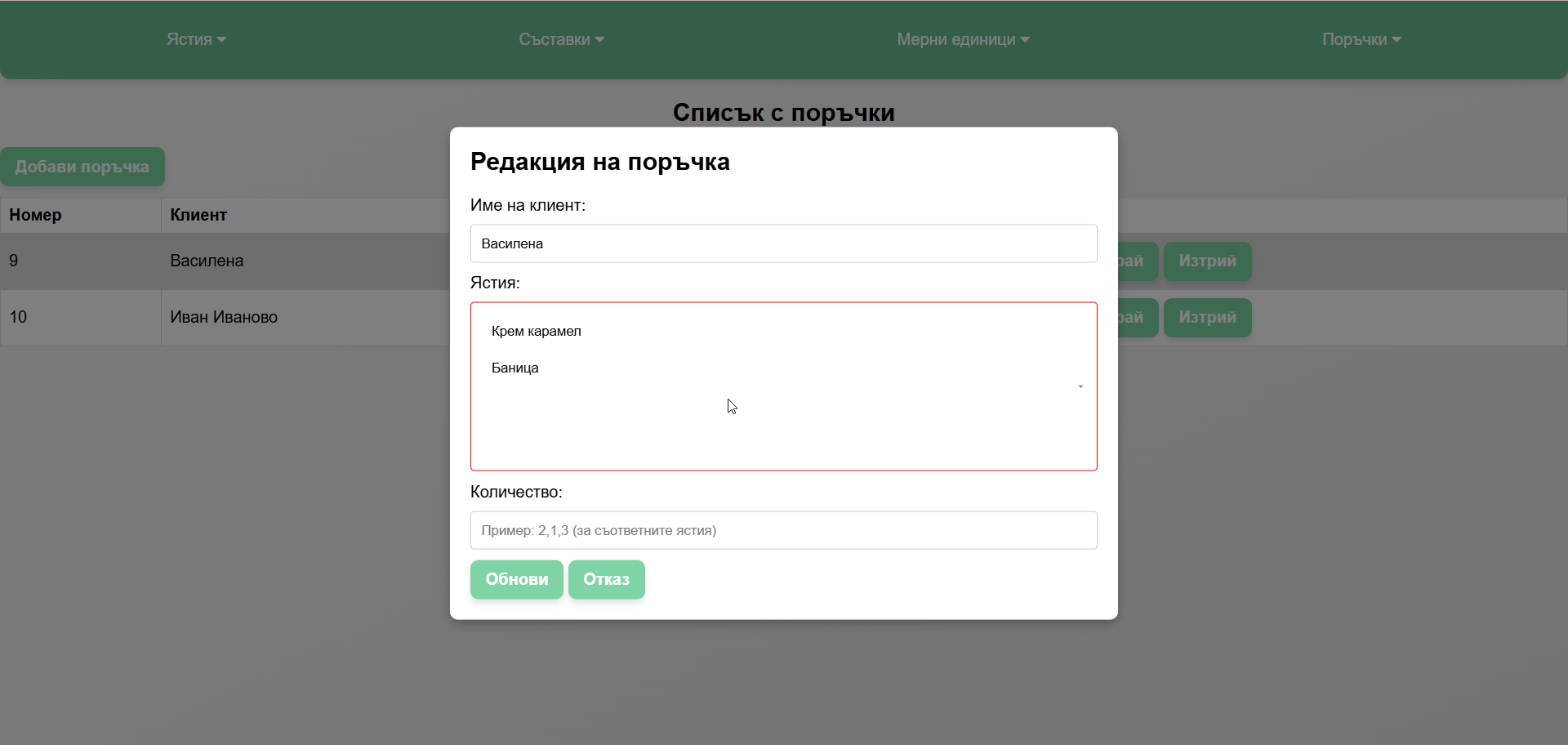
* Съобщение на успех или грешка



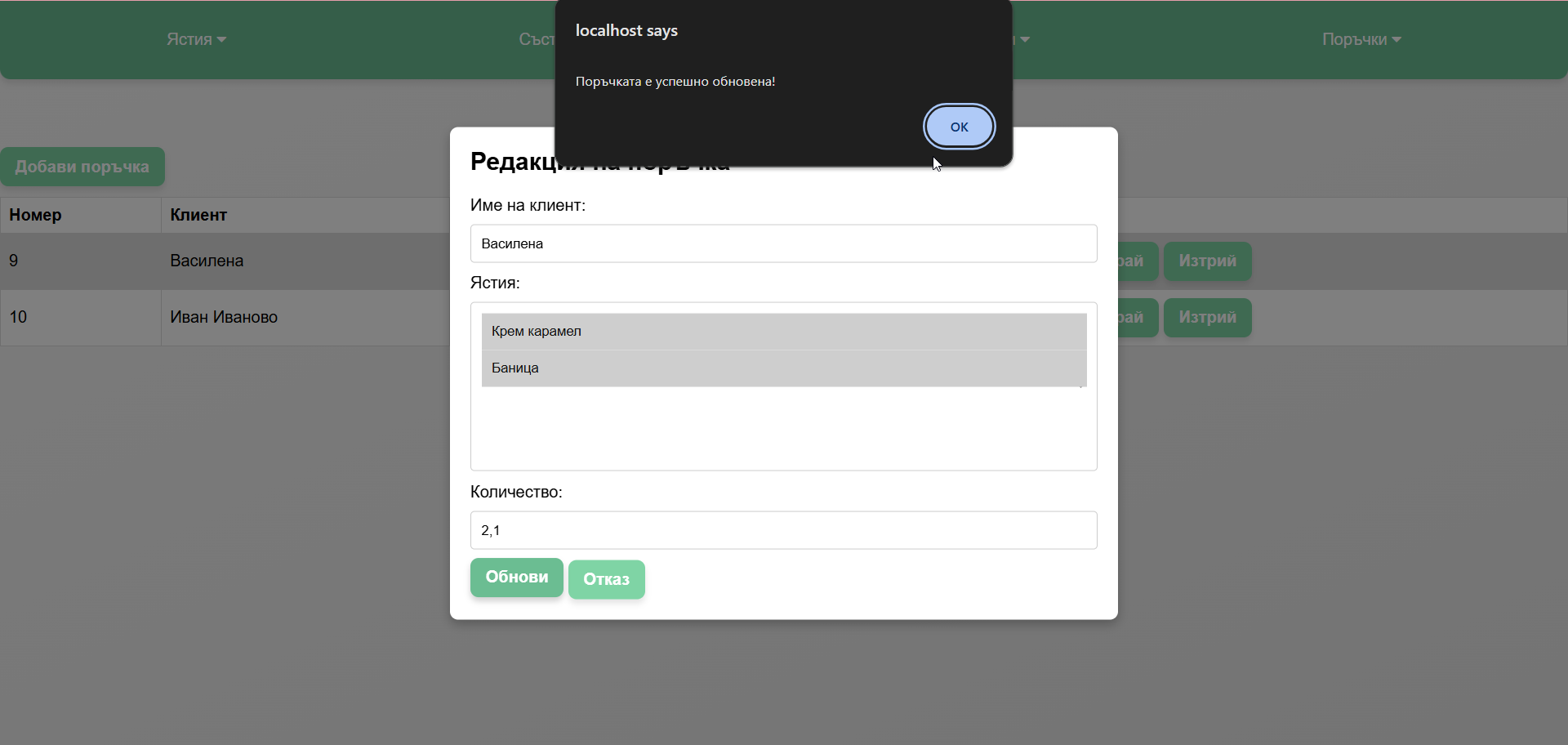
* Автоматично обновяване на таблицата след добавяне



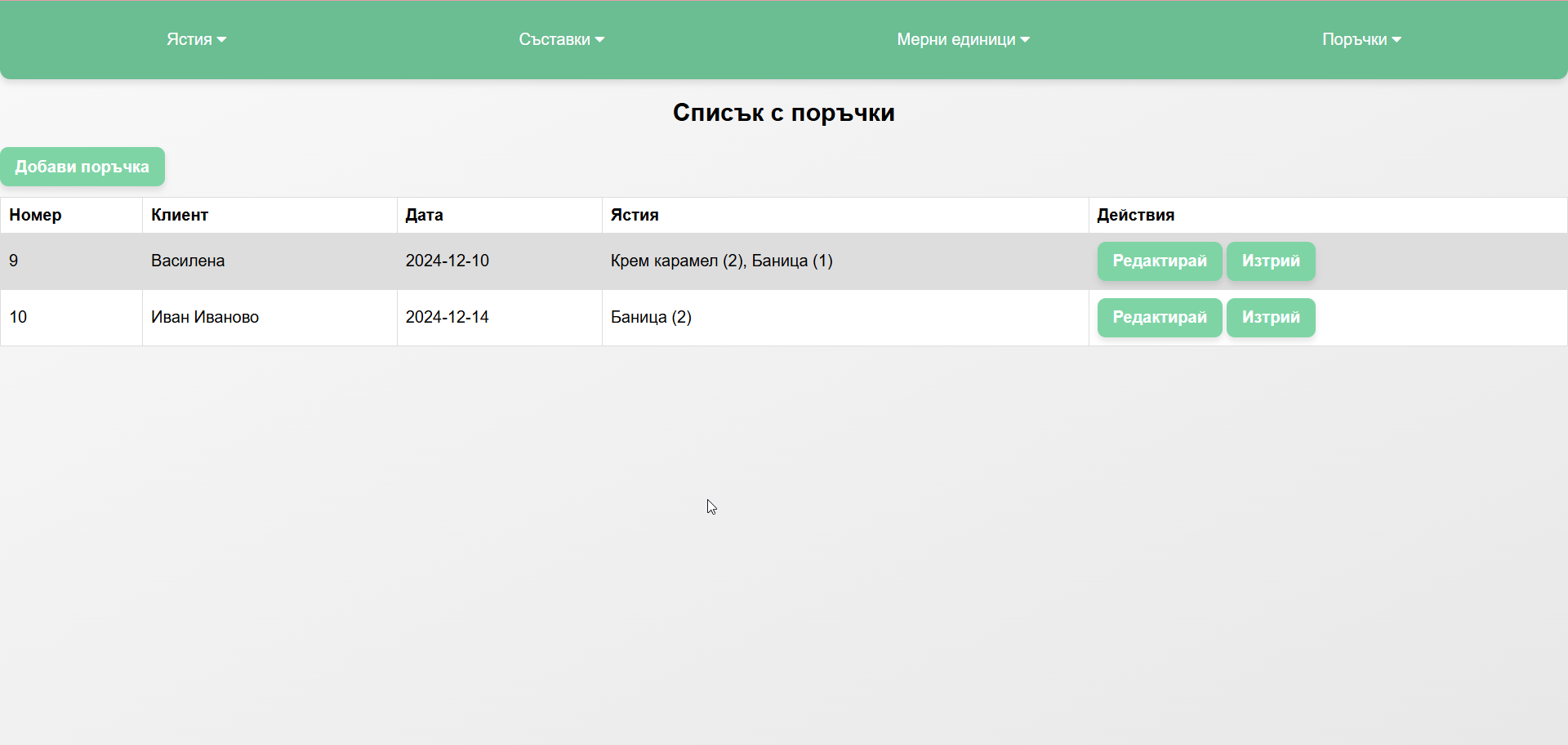
* Редакция
* Прозорец за редакция след избор на ред от таблица. Тук може да се подобри, тъй като не зарежда избраните вече ястия.



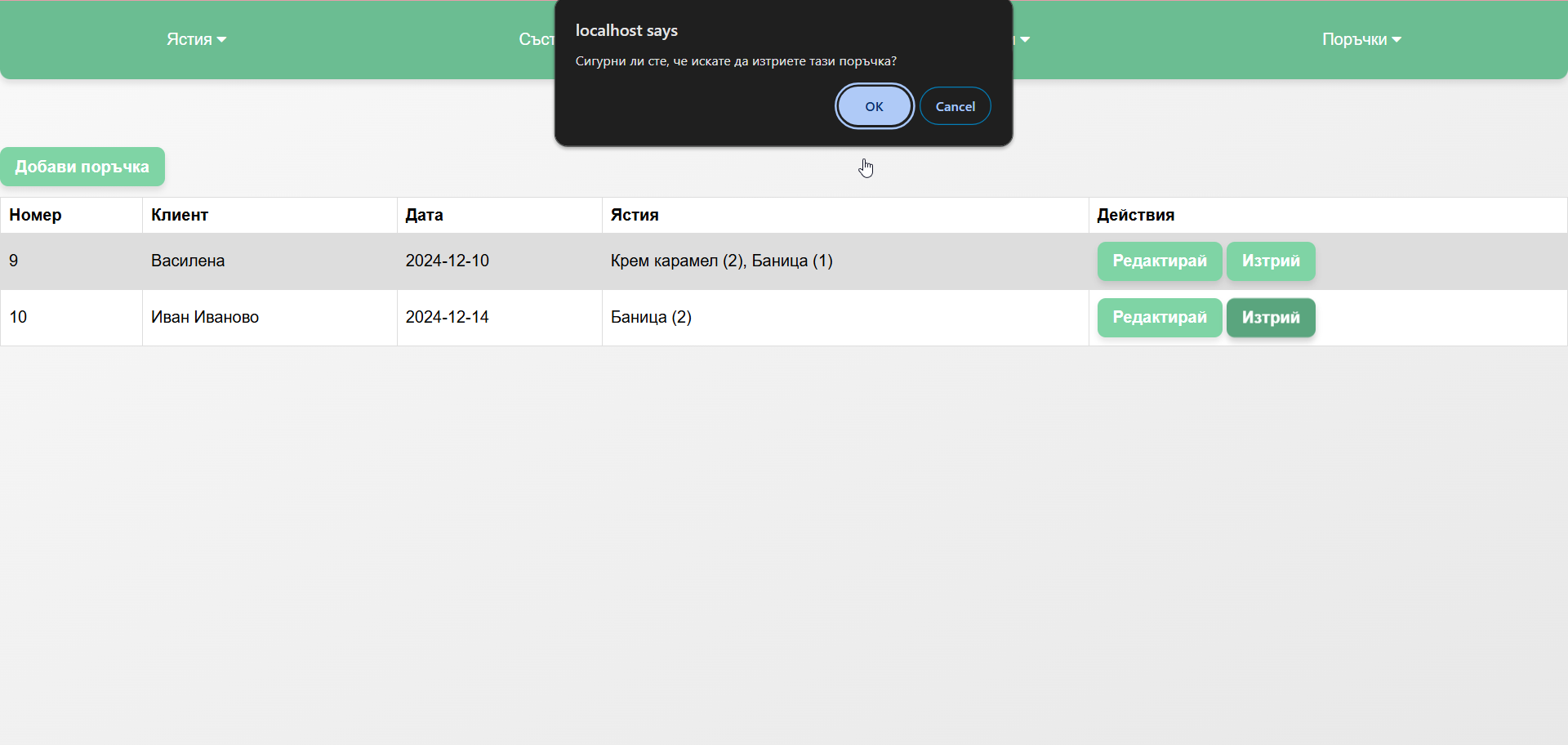
* Съобщение за успех или грешка



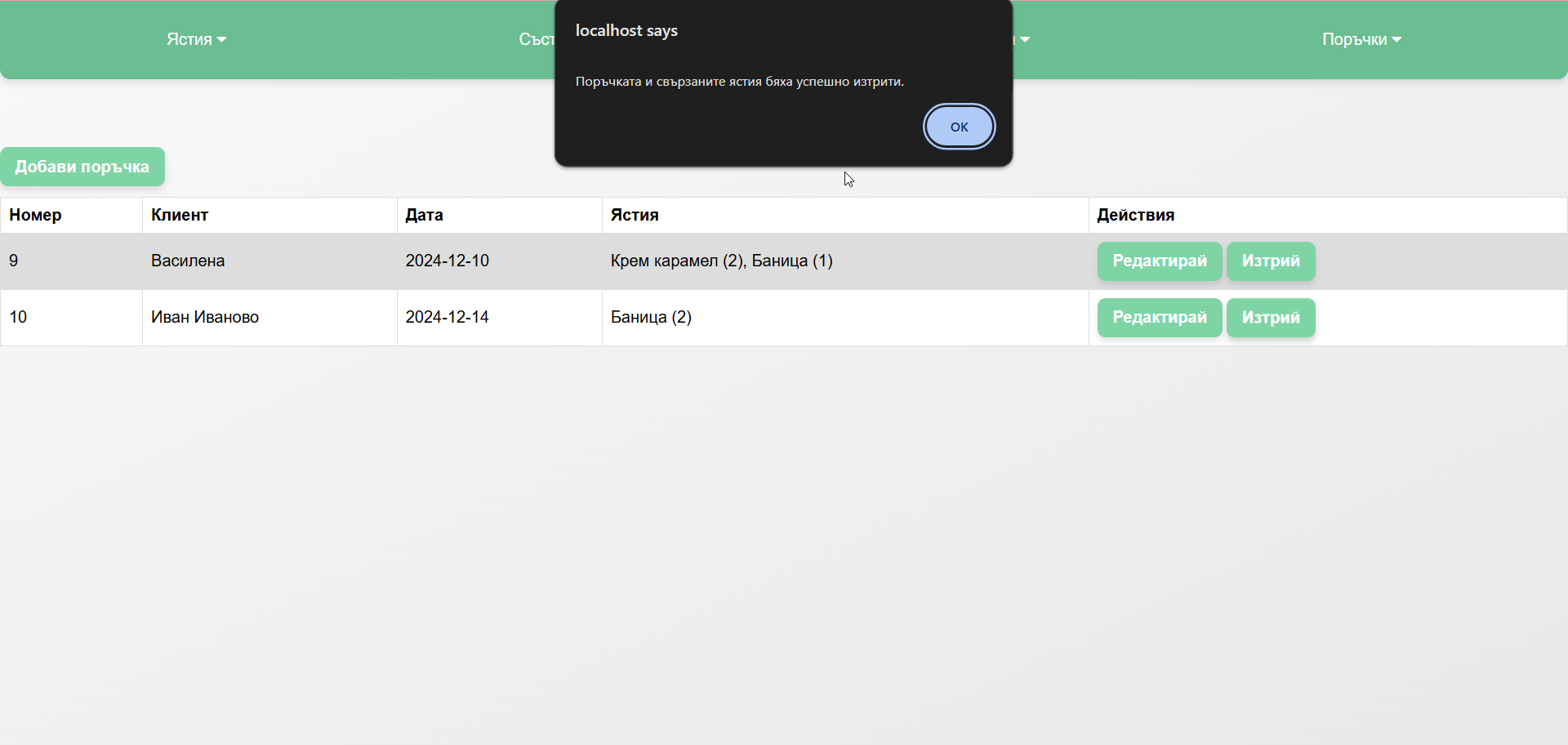
* Автоматично обновяване след редакция



* Изтриване
* Съобщение за изискване на съгласие



* Съобщение за успех или грешка



* 1. **Изводи и възможност за развитие**
* Всички модули функционират коректно.
* Възможности за подобрение на съществуващите функции
* Както пише в точките по – горе може да се подобри зареждането на информацията в падащите менюта и менютата с множество избори.
* Също може да се коригира изтриването при мерните единици с проверка, ако има мерна единица в съставка.
* Възможности за подобрение на приложението
* Добавяне на администраторска и клиентска част
* Добавяне на повече справки и автоматизация на обновяване на количествата
* Добавяне на ресторантско меню като информация за клиента какво предлага ресторанта
* Добавяне на повече валидации с цел защита на данните
* Добавяне на по – голяма сигурност отново с цел защита на данните
* Добавяне на функции за касови бележки и счетоводен софтуер за продажби

1. **Използвана литература**
   1. MySQL Reference Manual за проектиране на бази данни.
   2. PHP Official Documentation за създаване на CRUD операции.
   3. Bootstrap документация за оформяне на интерфейса.
2. **Списък на програмни компоненти**
   1. **Компоненти:**
      1. config.php: връзка с базата данни.
      2. created.php създаване на базата данни.
      3. ingredients.php: управление на съставки.
      4. orders.php: управление на поръчки.
      5. dishes.php: управление на ястия.
      6. Measures.php: управление на мерни единици.
      7. add\_unit.php: добавяне на мерна единица.
      8. update\_unit.php: редакция на мерна единица.
      9. add\_order.php: добавяне на поръчка.
      10. update\_order.php: редакция на поръчка.
      11. add\_dish.php: добавяне на ястие.
      12. update\_dish.php: редакция на ястие.
      13. add\_ingredient.php: добавяне на съставка.
      14. update\_ingredient.php: редакция на съставка.
      15. availability\_report.php: страницата за показване на наличните съставки.
      16. delete\_unit\_type.php: изтриване на мерна единица.
      17. delete\_dish.php: изтриване на ястие.
      18. delete\_order.php: изтриване на поръчка.
      19. delete\_ingredient.php: изтриване на съставка.
      20. menu.php: навигационния бар, който е в всяка страница.
      21. index.php: начална страница.
      22. navbar.css: дизайн на навигацията.
      23. tableandbuttons.css: дизайн на таблиците и бутоните.