

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА
ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ
Катедра „Компютърни науки и технологии“



ДИПЛОМНА РАБОТА

за придобиване
на ОКС „Бакалавър“

Тема:

„Разработка на Уеб-система за управление на задачи“

Изготвил: Лора Михайлова Ангелова

Специалност: Софтуерни и интернет технологии

Факултетен номер: 61162104

ТУ Варна, 2015 г.

Ръководител: доц. д-р инж. А. Антонов

Съдържание:

1. Въведение- необходимост от решаване на дипломната задача	4
1.1 Технически средства за изграждане	5
1.1.1 Основни моменти в изграждането на приложението	6
1.1.2 Сървърна част-PHP, MySql.....	6
1.1.3 Клиентска част	7
1.2 Постановка на дипломното задание	8
2. Въведение- теоретична част	9
2.1 Методи и средства за управление и планиране на задачи	9
2.1.1 Видове зависимости между дейностите	10
2.1.2 Мрежови графици.....	10
2.1.3 Диаграма на Гант	11
2.2 Основни подходи за реализация на преносим потребителски интерфейс чрез Java Script чрез библиотеки	12
3. Описание на програмното решение.....	15
3.1 Структура на приложението, описание на функциите на приложението, обосновка на взетите имплементационни решения и настройки.	15
3.1.1 Начална страница	15
3.1.2 Страница за регистрация	18
3.1.3 Страница за вход	20
3.1.4 Потребителска страница	20
3.1.4.1 За днес	22
3.1.4.2 Нова задача	26
3.1.4.3 Управление на задачи	27
3.1.4.4 Диаграма на задачите	30
3.1.4.5 Страница за добавяне на приход/разход.....	32
3.1.4.6 Управление на плащания	33
3.1.4.7 Диаграма на плащанията.....	36
3.1.7.8 Раздел с настройки на профила	38
3.2 Описание на сървърната част на клиентското приложение, описание на функциите и употребата на технологиите. Обосновка	39
3.2.1 Структура на база данни	39

3.2.2 Връзка с база данни	41
3.2.3 Регистрация	42
3.2.4 Вход в системата.....	42
3.2.5 Добавяне на записи	44
3.2.6 Обновяване на записи	46
3.2.7 Изтриване на записи	47
3.2.8 Изход.....	47
4. Документации	48
4.1 Използвани функции на Java Script библиотеката	48
4.1.1 Формуляр	48
4.1.2 Страничен раздел	52
4.1.3 Оформление.....	54
4.1.4 Таблици.....	55
4.1.5 Диаграми.....	58
4.2 Ръководство за потребителя.....	59
5. Заключение, оценка, тестване, предложения за развитие.	67
Приложение Б: Листинг на програмното осигуряване	

1. Въведение- необходимост от решаване на дипломната задача

В последните години все повече се използва глобалната мрежа. Масово се създават уеб приложения с всякакво съдържание и предназначение. С напредването на технологиите , нарастват и изискванията за изграждане на качествен потребителски интерфейс. Графичният интерфейс в наши дни е почти задължителен за всяко приложение, тъй като улеснява работата на обикновения потребител, който предимно работи с мишка.

Разглеждайки на базово ниво структурата дори на най-елементарен графичен компонент, се оказва, че зад визуализацията му се крие не прост код, определящ местоположението, формата, позицията и функциите му. Елементарни на пръв поглед операции като управление на бутон например се свеждат до различни подоперации като премигване при посочване с мишка, промяна на изображението при натискане на контролата, промяна при отпускане на бутона, промяна индикираща избран/ неизбран бутон и много други.

Хубавото при създаването на код за графичен интерфейс на приложение е, че има огромно количество преповтарящи се и подобни действия, които биха могли да се съхранят и преизползват многократно. С цел да се разграничи програмната от интерфейлната логика програмисти от цял свят вече създават голямо разнообразие от библиотеки за изграждане на качествен потребителски интерфейс. Това улеснява до голяма степен разработката на уеб приложенията, тъй като на уеб дизайнерите не се налага да пресъздават облика на приложението от основи, губейки часове за създаване на стилове, позициониране и планиране. Есествено, в някой случай се изискват по-специфични операции и облик, но дори тогава повечето библиотеки предлагат обстойна документация, която дава възможност на програмиста да манипулира кода и по този начин да персонализира компонентите и изгледите, които създава.

При работа с по-сложен интерфейс и наличието на повече функционалности е важно да се спазва определен стил, защото в противен случай, вместо интерфейсът да улесни работата и навигацията на потребителя, би се получила смесица от стилове и контроли, която излишно ще усложни използването на приложението.

Приложението разработено в дипломната работа има за цел да демонстрира разработката на цялостна уеб система, чиито интерфейс е изграден чрез използването на Java script библиотека.

1.1 Технически средства за изграждане

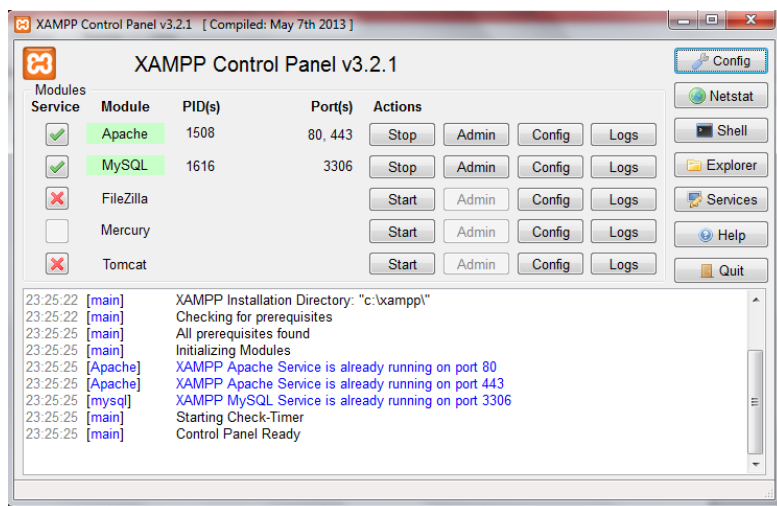
Изграждането на уеб система изисква доста средства, тъй като направата и съчетава в себе си много технологии.

Среда за програмиране (IDE- Interactive Development Environment)

NetBeans е среда за програмиране с безплатен и отворен код, която притежава инструменти за профилиране и разрешаване на пролеми. Тя предлага динамична поддръжка на езици като PHP и JavaScript, чрез автоматично допълване на кода и откриване на грешки:

Настройка на проекта

Хостинг- хостингът е локален и се извършва с помощта на XAMPP.



XAMPP е безплатен и свободен софтуерен пакет, който съдържа Apache HTTP Server, MySQL база от данни и други необходими инструменти за използване на езиците за програмиране PHP (с PEAR) и Perl. Програмата се разпространява с GNU General Public License.

Името на XAMPP е акроним от X (коя да е от четирите операционни системи), Apache, MySQL, PHP и Perl.

Чрез XAMPP се стартират Apache и MySQL.

Apache HTTP Server или само Apache е уеб сървър с отворен код, който има ключова роля за първоначалното разрастване на WWW. Чрез него работят над 70% от сайтовете. Той е необходим в разработката на уеб приложението, тъй като предоставя уеб съдържанието да бъде достъпно през интернет.

1.1.1 Основни моменти в изграждането на приложението

База данни

За създаването на базата от данни се използва phpMyAdmin, който се предоставя от пакетът за хостинг. В случая това е XAMPP. phpMyAdmin е безплатен софтуер, написан на PHP, създаден да поддържа администрацията на MySQL във Web. phpMyAdmin поддържа обширно множество от операции на MySQL.

Използвани технологии:

1.1.2 Сървърна част-PHP, MySql

PHP

PHP е безплатен, многоплатформен, скриптов, сървърен език за web програмиране, който може да се вгражда в HTML и позволява разработката на динамични Web сайтове. Към момента е налице PHP 5.6.4

Защо PHP ?

- PHP има отворен код;
- платформено независим;
- оптимизиран по отношение на времето за отговор;
- естествено вграждане в HTML страница;
- работа от страна на сървъра;
- простота на скриптирането;
- вградена възможност за връзка с множество бази данни;
- естествена работа с файловата система на сървъра;

- вградени възможности за взаимодействие с други сървъри на машината: LDAP, пощенски;

Предимства на сървър-базираните страници

- Сървърите обикновено са по-бързи и по-мощни от машините, използвани от средния Web клиент. Следователно, сървър-базираните динамични Web страници могат да разполагат с по-големи ресурси и да предоставят по-голямо потенциално съдържание на страниците.
- Обработката от страна на сървъра е единственият начин за съхраняване и извличане на информация в и от база данни. Данните могат да се организират по-добре и да се съхраняват непрекъснато във файлова система или релационни бази от данни, отколкото това е възможно с използването на клиент-базираните скриптове. Това е особено важно за по-големите сайтове, които имат множество страници и използването на база от данни може да намали значително разходите по поддържане на сайта.
- Обработката от страна на сървъра е независима от браузера. Сървърите изпращат обикновен HTML код и по този начин се избягват проблемите със съвместимостта с различните типове браузъри.

MySql

MySQL е многопоточна, многопотребителска, SQL система за управление на бази данни (СУБД) с повече от шест милиона инсталации.

Достъпни са програмни интерфейси, позволяващи програми, написани на различни програмни езици да имат достъп до MySQL бази данни. Такива са: C, C++, C#, Delphi (чрез dbExpress), Eiffel, Smalltalk, Java (с директна поддръжка), Lisp, Perl, PHP, Python, Ruby, REALbasic (Mac), FreeBasic, и Tcl, като всеки от тях има специфичен програмен интерфейс. Интерфейс тип ODBC наречен MyODBC позволява на други програмни езици, които поддържат ODBC интерфейс да комуникират с MySQL база данни, например: ASP или Coldfusion. MySQL е написан основно на ANSI C.

1.1.3 Клиентска част

Това е частта от приложението, която се изпълнява на клиентската машина. Езиците на които е написана са:
HTML, CSS, JavaScript (dhtmlxSuite библиотека)

HTML HyperText Markup Language е основният език за описание и дизайн на уеб страници. Документите се описват чрез HTML елементи, които се състоят от етикети и тагове и се заграждат с ъглови скоби. HTML елементите са основната градивна единица на уеб страниците.

В приложението HTML се използва за изграждане на интерфейсната част – формуляри, бутони. Данните се предават към сървърната част за обработка чрез методи POST И GET.

CSS (Cascading Style Sheets) е език за описание на стилове. Най-често се използва заедно с HTML, но може да се приложи върху произволен XML документ. Официално спецификацията на CSS се поддържа от W3C.

В проекта CSS се използва за описание на стиловете на някои от HTML документите и от библиотеката dhtmlx, с помощта на която е изградена клиентската част.

JavaScript –програмен език, който позволява да се променя динамично поведението на брауъра в рамките дадена HTML страница.

JavaScript се зарежда, интерпретира и изпълнява при клиента-от уеб брауъра, който му осигурява достъп до обектния модел на брауъра.

JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). Javascript е най-широко разпространеният език за програмиране в Интернет. Прието е JavaScript програмите да се наричат скриптове.

В проекта JavaScript намира приложение като основен език , използван в dhtmlx библиотеката.

1.2 Постановка на дипломното задание

Дипломното задание представлява уеб система за управление на задачи и плащания. Няколко са основните момента при разработката на системата.

Създаване на потребител.

За да се предостави подобна функционалност е необходимо да бъдат разграничавани отделните потребители и да им бъде предоставяно собствено пространство.

В проектът това е реализирано чрез релационни бази данни. Потребителите са уникални записи в таблица , а техните задачи и плащания се съхраняват в други таблици. Чрез външен ключ- идентификаторът на потребителя се разграничават записите, които принадлежат към него (1:n).

Първоначалният запис за нов потребител се извършва чрез предоставяне на регистрационна форма, и прихващане на потребителските данни , чрез изпращане на асинхронна AJAX заявка. След това те биват верифицирани и съхранени в базата.

Работа с приложението

Работа може да се извършва само след вход на потребител. Това позволява на приложението да съхрани сесийни променливи като идентификатор, име, презиме и настройки. С помощта на идентификатори се вземат записите от базата, принадлежащи на потребителя.

Тези данни се съхраняват на сървъра и с помощта на функции на езика се съставят xml файлове, които са необходими за функциите на JavaScript библиотека, чрез която е изграден интерфейсът.

Освен входни потребителски данни функциите на JavaScript библиотеката използват и други json файлове, които определят структурите на формулярите.

Персонализация на профилната страница.

За да се демонстрира използването на Java Script библиотеката dhtmlx с параметри съхранени в база данни се предоставя страница с настройки на потребителя. Настройките се съхраняват в таблицата за потребители , като основно това са низове, определящи цветовете.

2. Въведение- теоретична част

2.1 Методи и средства за управление и планиране на задачи

Много често в ежедневието се налага да се изпълнят няколко задачи и когато техния брой нарастне, а времето или финансите са ограничени е необходимо да се планира точното време, начин и последователност, в която те ще бъдат изпълнени.

Понякога правилното планиране е от изключителна важност, тъй като някои от действията могат да бъдат зависими от други, изпълнението на които може да забави, дори да прекрати цялостен проект.

2.1.1 Видове зависимости между дейностите

Съществуват няколко вида времеви зависимости между отделните задачи.

- Свърши, за да започне - това е най-често срещаната зависимост. За да започне задачата е необходимо дуга да е приключила. Пример: за да се използва приложението е необходимо да се извърши регистрация.
- Свърши, за да свърши.
- Започне, за да започне. Пример: за да започне процесът по контрол на качеството от страна на клиента на даден софтуер, същинската работа по самия продукт трябва също да започне. В противен случай клиентът няма какво да контролира.
- Започне, за да свърши. Пример: за да могат изпълнителите на един строителен обект да считат, че са си приключили работата, трябва приемно-предавателният процес да започне, за да може клиентът да дойде и направи оглед. Ако клиентът има забележки, значи те не са си свършили работата и трябва да оправят елементите, които не са според предварителните договори.

Най-елементарният вариант за управление на задачи е като бъдат съхранени и визуализирани. Така лесно може да се прецени техният брой. След приключването на дадена задачата отпада и вниманието се прехвърля към останалите.

Управлението зависи от няколко фактора, като приоритет на задачата, времетраене, време на започване, време на приключване, сложност и др. Функционалности като филтрация, промяна на приоритет, промяна на статус могат значително да улеснят управлението им. Когато става дума за сложен проект, може да говорим за подходи като мрежови графики и диаграми на Гант.

2.1.2 Мрежови графики

При проекти, които са съставени от множество задачи е необходимо да се използва мрежов график.

Мрежовият график наричан още диаграма на последователностите, поток на задачите или план на дейностите е инструмент, който се използва при управление на проекти. Целта на тази диаграма е чрез изобразяването на дейностите/заданията и връзките между тях под формата на мрежа/насочен граф да може да се придобие по-пълна картина за работата по проекта и по-точно планиране на времето и изграждане на графика.

Всяка задача се представя като възел на графа, а връзката между две задачи- чрез стрелка. Посоката на стрелката показва последователността. Този, възел, който сочи следва да се изпълни след този, от който излиза.

Мрежовите графици носят информация за последователността и продължителността на дейностите включени в проекта.

Мрежовите графици се използват основно когато две или повече дейности могат да бъдат извършени паралелно и се прилагат по-рядко при дейности разпределени последователно във времето.

Съвкупността от възли и стрелки представлява граф. Тъй като в графа са обозначени продължителностите на всяка една от дейностите можем да изчислим общата продължителност на отделните пътища представени в графа. Продължителността на даден път е съвкупността от продължителността на задачите, през които той минава. Мрежовият график носи още информация за последователността на отделните дейности и кои дейности могат да бъдат изпълнени паралелно.

Критичен път е най-дългият път между начален и краен възел на един мрежови график. Критичният път е определящ за общото времетраене на проекта и промени в неговата продължителност се отразяват върху общата продължителност на проекта.

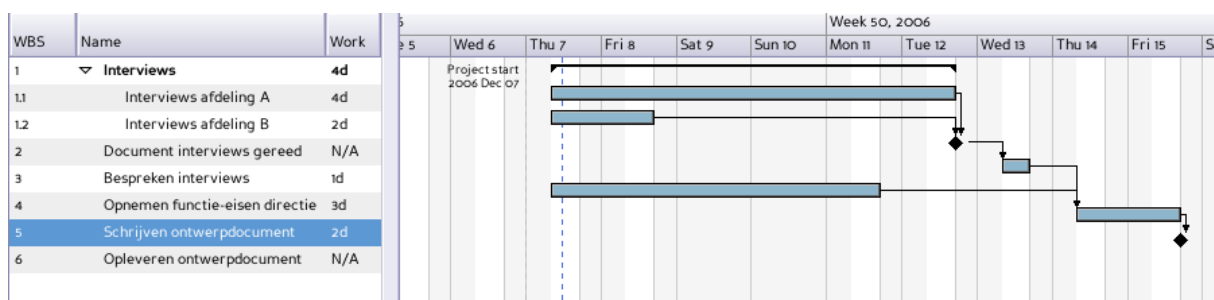
Когато се налага времетраенето на даден проект да бъде сведено до минимум, е необходимо да се оптимизират дейностите, включени в критичния път.

Основни предимства на мрежовите графици са, че те показват връзките между отделните дейности, тяхната последователност и продължителност и дават възможност за планиране във времето и оптимизация.

2.1.3 Диаграма на Гант

Диаграмата на Гант е кръстена на откривателя и Хенри Гант. Тя е вид диаграма с дейност на върха и е най-предпочитаният начин за разглеждане на един проект, защото визуално представя и времетраенето на отделните задачи. Дейностите в диаграмата на Гант са наредени хоризонтално – Всяка дейност е на един ред, а вертикалните линии разделят времето и обозначават даден момент. Всяка задача се отбелязва с правоъгълник, чиято дължина е пропорционална на времето за изпълнението и. По този начин се визуализира разположението на всички задачи във времето и е лесно да се види тяхното припокриване. Стълбовиден вид на диаграмата показва, че задачите са разпределени последователно във времето, а когато са една под друга става дума за паралелни действия.

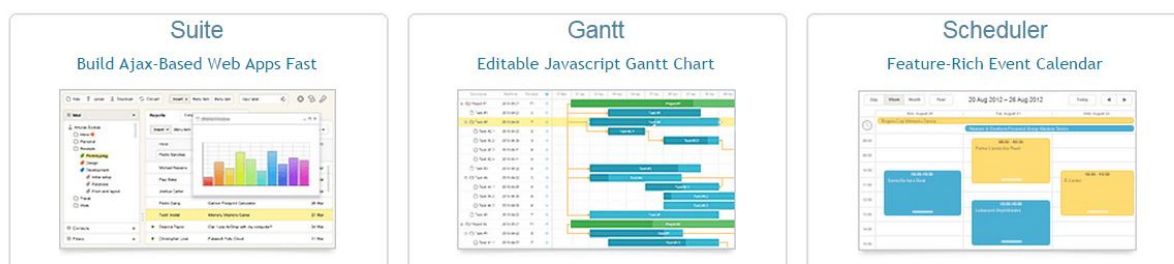
Впоследствие в диаграмите на Гант се налагат и специфични означения за дейности (своеобразни скоби), които са съставни, както и за ключови дати (малки ромбчета).



Диаграма на Гант, както изглежда в продукта Gnome Planner

2.2 Основни подходи за реализация на преносим потребителски интерфейс чрез Java Script чрез библиотеки.

DHTMLXSuite



Същност

DHTMLX е библиотека, която притежава всички необходими свойства за разработката на потребителски интерфейс. Разработва се от години наред и има хиляди потребители по целия свят. Поради факта, че лесно се разучава, че има инуитивен интерфейс и множество подробни ръководства DHTMLX позволява да се разработват интерактивни уеб приложения по-бързо и с по-малко усилия.

DHTMLX пакетът включва повече от 20 професионално проектирани, персонализируеи компоненти за потребителски интерфейс, който покриват почти всички нужди на уеб приложенията като таблицим контроли с автоматично попълване и други.

Структура на пакета

- **codebase** – пакетирани файлове с код на библиотеката. Тези файлове имат много малък размер с цел заемане на по-малко пространство и по малък трафик при работа в интернет. В приложенията е необходимо да се използват файловете от следните папки:
 - **ext** – папка, която съдържа добавки за обновяване на компонентите на формулярите
 - **thirdparty** – папка, която включва third-party библиотеки за коректната работа на Chart компонентите
 - **imgs** – папка съхраняваща изображения, които са ресурси необходими за играждането на компонентите;
 - **dhtmlx.js** – скриптов файл;
 - **dhtmlx.css** – файл за стилизиране;
 - **dhtmlx_deprecated.js** – файл, съхраняващ API от предните версии;
- **docs** – папка съхраняваща документацията.
- **samples** – примерни за използване на компонентите;
- **skins** – папка с различни теми ;
- **sources** – файловете с кодовете на библиотеката. Не са минимизирани с цел лесно четене. Използва се при разрешаване на проблеми:
 - **libCompiler** – инструмент, който позволява множество от файлове да се компилират като индивидуален скрипт.
 - **dhtmlxWindows ... dhtmlxAccordion** – папки с файлове с код за индивидуалните компоненти
- **License_gpl_2.0** – файл с информация за лиценз

DHTMLXDataStore осигурява съхранение за всички DHTMLX компоненти, които се използват в уеб приложенията. С DataStore могат да се визуализира една и съща информация в различни компоненти- таблици, дървета, формуляри и други. Всички операции с данните- добавяне, премахване , филтриране могат да се извършват чрез DataStore, който позволява да се прилагат промени за всички интерфейсни компоненти.

Интеграция със сървър

DHTMLX е библиотека, която се използва в клиентската част и може да се интегрира с всяка сървърна технология. Единственото изискване за интеграцията е точен формат на данните, които интерфейсните компоненти използват. Данните се съхраняват в формати като XML, JSON, SCV и т.н.

За опростяване на интеграцията със сървърната част, се предлагат конектори Connectors за PHP и ASP, NET, Java и ColdFusion. Възможно е да се използва и скриптът node.js , за включване на автоматични обновявания.

В случаите, когато имамем собствен сървърен скрипт за обработка на данни, както в проекта са разработени методи, които изпращат данните към сървъра чрез AJAX POST/GET заявки. По подразбиране се използва POST.

```
myForm.attachEvent("onButtonClick",function(id){
    if(id=="send_button"){
        myForm.send("php/save_form.php", "get", function(loader, response){
            alert(response); });
    }
});
```

Също така може да се използва HTML формулярно потвърждение

```
<form action="php/save_form.php" method="post" target="[some target]">
    <div id="form_container"></div>
</form>
<script>
    myForm = new dhtmlXForm("form_container",structureAr);
    myForm.attachEvent("onButtonClick",function(id){
        if(id=="my_button"){
            if(myForm.validate())
                document.forms[0].submit()
        }
    })
</script>
```

3. Описание на програмното решение

3.1 Структура на приложението, описание на функциите на приложението, обосновка на взетите имплементационни решения и настройки.

Приложението има за цел да демонстрира разработката на уеб-базирана система, чиито интерфейс е изграден с помощта на Java Script библиотека. Състои се от клиентка и сървърна част. Клиентската част е създадена чрез езиците Java Script, HTML и CSS.

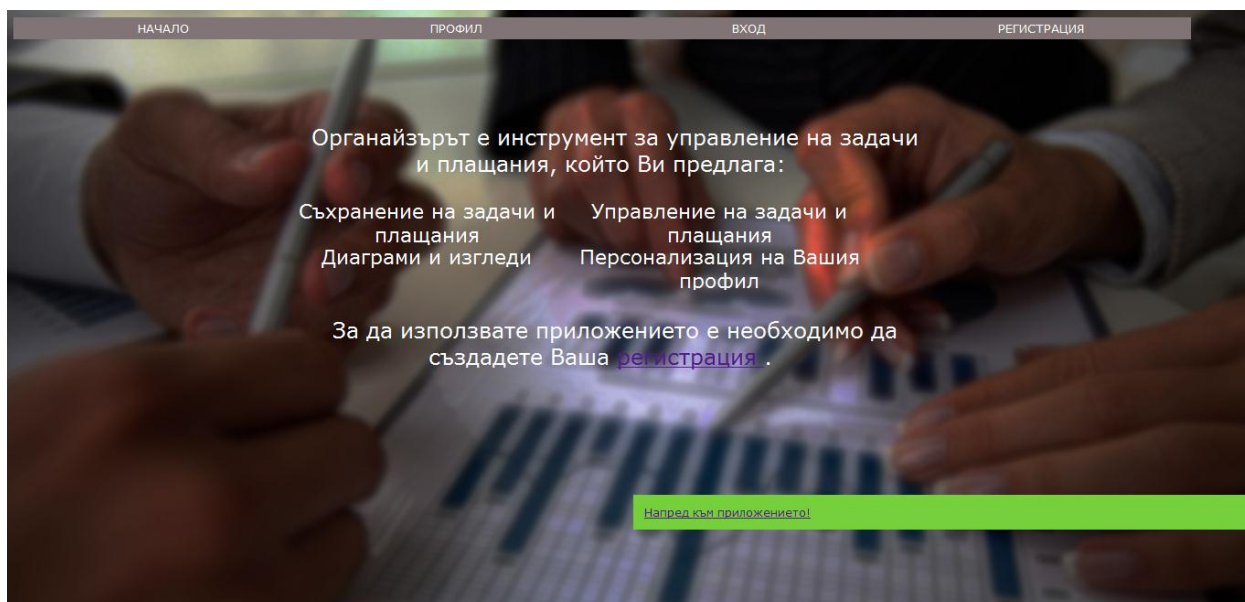
Основни папки с файлове, на проекта:

- Codebase- в тази папка се намират всички скриптове, който библиотеката DHTMLX използва
- Common-в тази папка се намират допълнителни ресурси, като изображения, които скриптовите използват за създаване на интерфейса.
- Css- тук се намират стиловете на някои от страниците- като logout.php
- Json-папка с файлове, които съхраняват структурите на dhtmlx компоненти като sidebar и form.
- Skins- в тази папка се намират стиловете за трите теми, които предлага пакетът DHTMLXSuite
- Xml- xml-файлове за всички таблици, които се визуализират.

Страници на приложението

- Начална страница
- Регистрационна страница
- Страница за вход
- Потребителска страница

3.1.1 Начална страница index.php



Тази страница се използва като индексна. Съдържанието и представлява кратка и пояснителна информация за приложението и неговите функции. Има няколко линка, като за главна навигация се използва меню.

Меню

```
<ul>
  <li id="liMenu"><a href="index.php">Начало </a></li>
  <li id="liMenu" ><a href="userPage.php" id>Профил </a></li>
  <?php if(isset($_SESSION['username'])) {
    echo  "<li id=\"liMenu\"><a href=\"logout.php\">Изход </a></li>";
    else{ echo"
  <li id=\"liMenu\" color=\"red\"><a href=\"login.php\" id> Вход
  </a></li>
  <li id=\"liMenu\" color=\"red\"><a
  href=\"registration_1.php\"id>Регистрация </a></li>";
  }
?>

</ul>
```

Менюто служи за навигация в системата. То представлява неномериран HTML списък.

Всеки елемент `` на списъкът съдържа линк към страница или функционалност. Тъй като не във всички случаи е необходимо да фигурират всички линкове е вграден php скрипт, който проверява дали навигацията се извършва след вход на потребител, или не. Това става с помощта на сесийната променлива `$_SESSION['username']`. Ако има установено

потребителско име, това значи , че няма нужда от линк за вход, а за изход. Ако няма установено име, тогава се налага да се включи възможност не само за вход, а и за регистрация, тъй като не се знае дали този, който в момента навигира е регистриран потребител.

За стилизиране на менюто според дизайна на сайта се използва вграден стил, който дефинира позиционирането, големината, цвѐта и шрифта на елементите на списъка (менюто).

```
<style>

    .basic {

        position: absolute;
        left: 300px;
        top:100px;
        width: 700px;
        font-family: Verdana, Geneva, sans-serif;
        font-size: 24px;
        color: white;
        text-align: center;

    }

    .right {

        position: absolute;
        right: 0px;
        top:530px;
        width: 680px;
        background-color: #75d03b;
        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;

    }

</style>
```

Останалите линкове в страницата са вградени по различен начин. За описанието на приложението, което се намира в централната част на страницата също се използва сървърен скрипт, проверяващ дали е настъпил вход. Ако не е, към текстът, който се визуализира, се добавя допълнително съобщение, което пояснява на потребителя, че трябва да е регистриран. Към съобщението непосредствено е добавен и линк, който се откроява с различен цвят и големина и при избор, отваря страницата с регистрационна форма.

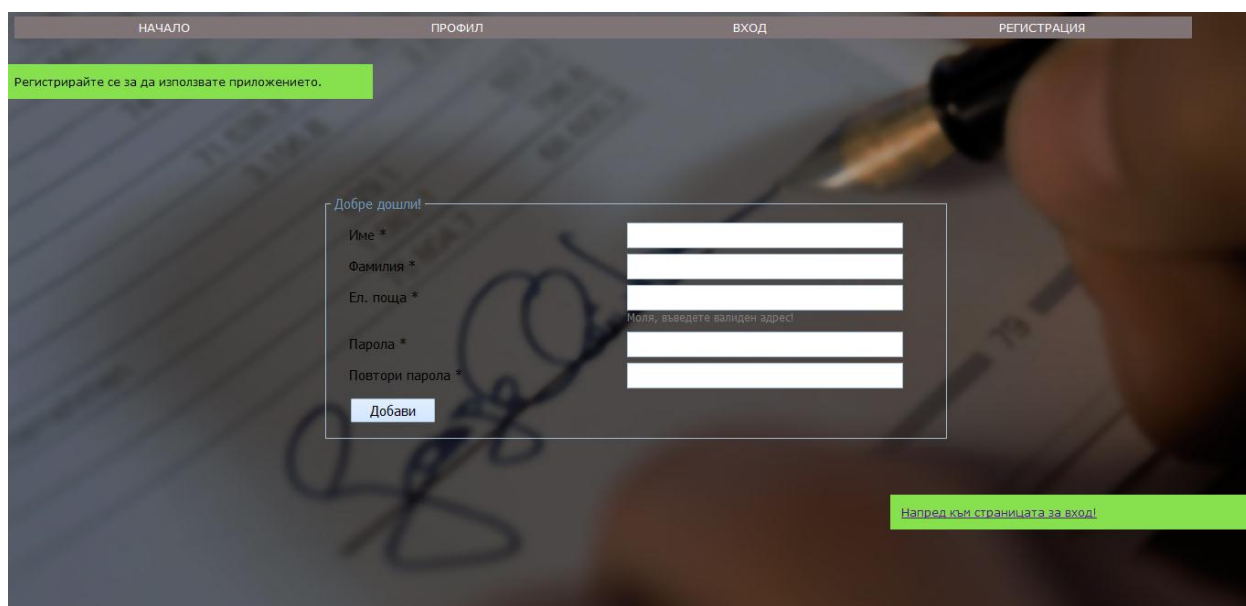
```
<?php if((!isset($_SESSION['username'])))
{
    echo" &nbsp; &nbsp; &nbsp; За да използвате приложението е необходимо
    да създадете Ваша <a href=\"registration_1.php\" > регистрация </a>.";
}??

<div class="right">
    <p> &nbsp; &nbsp; &nbsp; <a href="userPage.php">Напред към приложението!</a></p>
</div>
```

Линкът „към приложението“ води към страницата на потребителя. До нея може да се достъпи и чрез менюто. Той също е стилизиран чрез вграден стил CSS. Специфира се по class selector (class="right">).

Умишлено този линк присъства в менюто дори когато не е налице вход, тъй като при отварянето му, потребителят ще бъде пренасочен към страницата за вход.

3.1.2 Страница за регистрация



На страницата за регистрация са поместени меню, формуляр, линк и пояснително съобщение. Пояснителното съобщение е се визуализира чрез параграф, в горния ляв ъгъл на страницата. Цветът, позицията големината и шрифта се задават чрез CSS стил.

Формулярът, намиращ се в средата на страницата се създава чрез скрипт, при инициализация с помощта на функцията на Java Script библиотеката dhtmlx.

```
myForm = new dhtmlXForm("myForm");
```

Обектът използва файл в json формат. Той определя структурата и съдържанието на формуляра.

```
myForm.loadStruct("json/registration.json", function() {});
```

registration.json

```
[
```

```
{type: "settings", position: "label-left", labelWidth: 300, inputWidth: 300},
{type: "fieldset", label: "Добре дошли!", inputWidth: 680, list:[
{type: "input",name: "tFirst", label: "Име *", validate: "NotEmpty"},
{type: "input",name: "tLast", label: "Фамилия *", validate: "NotEmpty"},
{type: "input", name: "tMail", label: "Ел. поща *",validate:
"NotEmpty,ValidEmail", note: {text: "Моля, въведете валиден адрес!"}},
{type: "password", name: "pPass", label: "Парола *", validate:
"NotEmpty"},
{type: "password", name: "pPass2", label: "Повтори парола *", validate:
"NotEmpty"},
{type: "button", name: "send" , id: "send", value: "Добави" }
]}
]
```

За да се извърши регистрацията е необходимо всички полета да са попълнени и съдържанието на полето за електронен адрес да е валиден електронен адрес. За тази цел се използват валидационните правила на библиотеките, които се указват в атрибутите на компонентите в Json файла.

```
validate: "NotEmpty,ValidEmail"
```

Валидирането се извършва при събитие “onclick” на бутона “Регистрация”.

При валидни данни и съответстващи пароли се изпраща AJAX заявка към сървърния скрипт `registerme.php`

```
myForm.attachEvent("onButtonClick", function (id) {
if (id == "send") {
    if(myForm.validate()){
        if(myForm.getItemValue("pPass")==myForm.getItemValue("pPass2")){
            myForm.send("registerme.php", "post", function (loader, response) {
                alert(response);
            });
        }
        else {
            alert("Паролите не съвпадат!");
        }
    }
}
});
```

Под бутона на формуляра за регистрация се намира параграф с линк към страницата за вход. Той е стилизиран чрез CSS.

3.1.3 Страница за вход login.php

НАЧАЛО ПРОФИЛ ВХОД РЕГИСТРАЦИЯ

Влезте в профила си, за да използвате приложението!

Електронен адрес :

Парола:

Вход

Страницата за вход служи за вход в системата. След него потребителят вече има право на достъп до останалите функционалности на потребителската страница.

На нея са поместени меню и формуляр. При успешен вход потребителят се пренасочва към потребителската страница. За проверката на данните отговаря сървърният скрипт.

3.1.4 Потребителска страница userPage.php

НАЧАЛО ПРОФИЛ ИЗХОД

Здравейте, Lora Angelova !

За днес

Добави нова задача

Управление на задачите

Диаграма на задачите

Нов приход/разходи

Управление на плащания

Диаграма на плащания

Настройки на профила

Задачи

Отбележи като приключена Премахване

НОМЕР	ДАТА	ЗАГЛАВИЕ	ОПИСАНИЕ	ЗАДЪЛЖИТЕЛНА	ПРИКЛЮЧЕНА
151	2015-06-28	Настройки на	Да се направи формуляр за настройки.	Да	Да
170	2015-06-28	Формуляри	Да се позиционират формулярите	Да	Не

Плащания

Отбележи като приключена Премахване

НОМЕР	ВИД	ДАТА	ОПИСАНИЕ	РАЗМЕР	ЗАДЪЛЖИТЕЛЕН	ПР
15	Разход	2015-06-28	Наем	100	Не	Не

Потребителската страница е изградена с помощта на HTML и CSS.

В горната и част е поместено меню и поздравително съобщение.

Дизайнът на потребителската страница се базира на компонента sidebar(страничен раздел) от dhtmlx пакета. Този компонент прави възможно поместването на няколко раздела с различно съдържание на една страница. Той предлага меню, към което могат да се добавят икони и текст. Превключването на различните точки отваря различни раздели.

```
mySidebar = new dhtmlXSideBar({
    parent: "sidebarObj",
    icons_path: "common/win_32x32/",
    template: "tiles",
    width: 300,
    json: "json/sidebar.json",
    onload: function () {../Съдържание.}
```

В конструктора на страничния бар се указват: път на иконите, които ще се използват, шаблон на визуализация, ширина, json файл със структурата на раздела и функция която да се изпълни при зареждане.

sidebar.json

```
{items: [
  {id: "today", text: "За днес", icon: "today.png", selected: true},
  {id: "addTask",text: "Добави нова задача", icon: "add.png"},
  {id: "manageTasks", text: "Управление на задачите", icon: "documents.png"},
  {type: "separator"},
  {id: "day", text: "Диаграма на задачите", icon: "chart.png"},
  {id: "addInOut",text: "Нов приход/разходи", icon:"addMoney2.png"},
  {id: "managePayments", text: "Управление на плащания", icon:"money.png"},
  {id: "calculate", text: "Диаграма на плащания", icon: "pie.png"},
  {id: "settings", text: "Настройки на профила", icon: "settings.png"},
]}
```

Страничният бар в приложението съдържа осем раздела. Всеки от разделите е предназначен за различна функционалност. Някои от тях използват подразделения. Това се осъществява чрез прикачването на layout (оформление). Когато имаме оформление за даден раздел, компонентите се прикачват към него. Така може в една част на страничния

раздел да разполагаме с няколко подразделения и така да подредим компонентите по лесно и прегледно.

Подраздели:

- За днес
- Добави задача
- Управление на задачите
- Диаграма на задачите
- Добави приход/разход
- Диаграма на плащанията
- Настройки на профила

3.1.4.1 За днес

В тази част се визуализират две таблици- за плащания и за задачи. Това са записите за текущата дата.

Първоначално се създава ново оформляние с два раздела- за задачи и плащания. Това става с помощта на dhtmlx компонента layout.

```
myLayoutToday = mySidebar.cells("today").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Задачи"}, {id: "b", text: "Плащания"}]
});
```

За да се визуализират записите се използва таблица. Тя се създава с dhtmlx компонента grid.

```
myGridTodayTasks = myLayoutToday.cells("a").attachGrid();
myGridTodayTasks.setImagePath("imgs/");
<?php include('getTodayTasks.php'); ?>
myGridTodayTasks.loadXML("xml/todayTasks.xml");
myGridTodayTasks.enableAlterCss("even", "uneven");
```

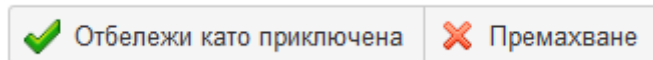
При създаването се указва в коя част на оформлението ще се помести таблицата. Данните за нея се съхраняват в база от данни. Извличането им става чрез php скрипт.

```
<?php include('getTodayTasks.php'); ?>
```

Той сформира xml файл със съдържанието и структурата на таблицата. След като се сформират входните данни, те се зареждат в таблицата.

```
myGridTodayTasks.loadXML("xml/todayTasks.xml");
```

За да могат да се управляват таблиците със записи, към всеки раздел се прикачва лента с бутони.



```
myMenuTodayTasks = myLayoutToday.cells("a").attachToolbar({  
    icons_path: "common/imgs/",  
    xml: "xml/toolbarToday.xml"  
});
```

Лентата с бутони е компонент на dhtmlx библиотеката, който има свойството да се вгражда в раздел на оформлението и в различни други компоненти. Нейната структура се съхранява в xml файл. За да се извършат някакви операции чрез бутоните е необходимо да се прикачи действие към събитие по формата. В случая това става на събитието “onclick”. За да се установи кой бутон е избран се гледа неговото id.

```
myMenuTodayTasks.attachEvent("onClick", function (id) {  
    if (id == "delete")  
    {  
        var selectionTodayTasks=myGridTodayTasks.getSelectedRowId();  
        var valueCellTodayTasks= myGridTodayTasks.cells(selectionTodayTasks,0  
        ).getValue();  
        deleteTask(valueCellTodayTasks);  
        myGridTodayTasks.deleteSelectedItem();  
        updateGrids();  
    }  
});
```

В случая лентата за задачи има два бутона:

- За промяна на статус на задачата- При натискане на този бутон се взима id на реда от таблицата, чрез функцията `getSelectedRowId()`. Чрез неговия номер се намира стойността на колоната за идентификатор на задачата. След това се вика съвършен скрипт, като чрез AJAX заявка му се изпраща id на задачата. По него той намира записа в таблицата с данни и обновява полето с нова стойност (`completed=true`).

```
function completeTask(id)  
{  
    $.ajax({
```

```

type: 'POST',
data: {"taskId": id},
url: 'setTaskCompleted.php',
success: function (response) {
    alert(response);
    $("#return").text("result from php " + response);
    updateGrids();
}
});
}

```

- За изтриване на задачата- `deleteTask()` . Изтриването на задача отново се извършва чрез AJAX заявка. Подават се `id` на указания в заявката сървърен скрипт. От там нататък той извършва необходимите операции с базата от данни за да изтрие задачата с подадения идентификационен номер.

```

function deleteTask(id)
{
    $.ajax({
        type: 'POST',
        data: {"taskId": id},
        url: 'deleteTask.php',
        success: function (response) {
            alert(response);
            $("#return").text("result from php " + response);
            updateGrids();
        }
    });
}

```

След всяка една от тези операции е необходимо обновяване на таблиците с нови стойности.

Таблицата за плащания се създава по аналогичен начин.

```

myGridTodayPayments = myLayoutToday.cells("b").attachGrid();
myGridTodayPayments.setImagePath("imgs/");
<?php include('getTodayPayments.php'); ?>
myGridTodayPayments.loadXML("xml/todayPayments.xml");
myGridTodayPayments.enableAlterCss("even", "uneven");

```


Скриптът, който извлича данните праща SQL заявка за таблицата incomes. Обектът използва готовия xml файл и създава таблица с плащания към днешна дата. Плащанията са два вида – приходи и разходи. Тъй като се визуализират в една таблица, логическото разделение се извършва чрез колона от таблицата, чиято стойност показва дали плащането е приход или разход.

Лентата към този раздел на оформлението отново има два бутона със същите функции. За изтриване на плащане и за смяна на статуса. Тъй като са два вида записа е нужно текстът на бутонът да се променя в зависимост от селектирания ред. За да се сменя текстът на бутона се добавя събитие при селектиране на ред onRowSelect.

```
myGridTodayPayments.attachEvent("onRowSelect", doOnRowSelectedTodayPayments);

function doOnRowSelectedTodayPayments() {
    var selectionTodayPayments = myGridTodayPayments.getSelectedRowId();
    var valueCellPayments = myGridTodayPayments.cells(selectionTodayPayments,
1).getValue();
    var myMenuTodayPaymentsItem;
    if (valueCellPayments == "Приход")
        myMenuTodayPaymentsItem = "Отбележи като получен";
    else{
        myMenuTodayPaymentsItem = "Отбележи като платен";
        myMenuTodayPayments.setItemText("mark", myMenuTodayPaymentsItem);
    }
}
```

3.1.4.2 Нова задача

Този раздел служи за добавяне на нов запис в таблицата usertasks. Интерфейсът е изграден с помощта на формуляр.

```
myForm = mySidebar.cells("addTask").attachForm();
myForm.loadStruct("json/form.json");
myForm.attachEvent("onButtonClick", function (id) {
    if (id == "send") {
        myForm.send("savingTask.php", "post",
function (loader, response) {
            alert(response);
            updateGrids();
            clearmyForm();
        });
    }
    if (id == "clear") {
        clearmyForm();
    }
});
```

Обектът на формуляра използва json файл, с който сформира структурата му.

```
[
{type: "settings", position: "label-left", labelWidth: 130, inputWidth: 300,
offsetLeft: 10},
{type: "label", label: "Нова задача"},
{type: "input", name: "title", label: "Заглавие *", validate: "NotEmpty",
offsetTop: 10},
```

```

{type: "input", name: "description", label: "Описание", rows: 10},
{type: "calendar", name: "date", dateFormat: "%Y-%m-%d", validate:
"NotEmpty", label: "Дата*", calendarPosition: "right"},
{type: "checkbox", name: "urgent", label: "Неотложна"},
{type: "button", id: "send", value: "Добави", name: "send"},
{type: "button", id: "clear", value: "Изчисти", name: "clear"}
]

```

Полетата се делят на задължителни и незадължителни.

- Заглавие – поле за заглавие на задачата, поясняващо нейното естество
- Описание- поле за подробно описание на задачата
- Дата- дата на задачата
- Неотложна- приоритет на задачата

За да се провери дали задължителните полета са попълнени се използват валидационни правила. След съхранение на задача се вика функция за почистване на полетата на формуляра. Зачистването им може да стане и преди съхранение на задачата, чрез натискане на бутон „Изчисти“.

3.1.4.3 Управление на задачи

Здравейте, Lora Angelova !

За днес

Добави нова задача

Управление на задачите

Диаграма на задачите

Добави приход/разход

Управление на плащания

Диаграма на плащания

Настройки на профила

За дата: Премахване Обнови

Задължителни Незадължителни Приключени Неприключени

НОМЕР	ДАТА	ЗАГЛАВИЕ	ОПИСАНИЕ	ЗАДЪЛЖИТЕЛНА	ПРИКЛЮЧЕНА
150	2015-06-19	Плащания	Да направя диаграмата	Да	Да
151	2015-06-28	Настройки на	Да се направи формуляр за настройки.	Да	Да
153	2015-06-27	Чекбуксове	Чекбуксове за масово изтриване!	Не	Не
156	2015-06-19	ИП	Ла скрия ИП	Ла	На

Преглед

Нова задача

Заглавие *

Дата

Описание

Неотложна ☐

Завършена ☐

Запази

Този раздел служи за унагледяване и управление на задачи.

Унагледяването става възможно чрез филтриране на съдържанието, а управлението чрез редакция и изтриване.

Таблицата съдържа всички задачи, за текущия потребител. Съдържа следните колони:

- Номер
- Дата
- Описание
- Задължителна
- Приключена

Тук лентата с бутони е прикачена към раздела.

```
myMenu = mySidebar.cells("manageTasks").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarUserPage.xml"
});
```

Към нея освен стандартните бутони има и по особена функционалност- динамично сортиране по дата. То има за цел да филтрира задачите само за избраната дата. Към обекта на лентата с бутони се добавя входно поле с прикачен календар

```
myMenu.addText("text_from", null, "За дата");
myMenu.addInput("date_from", null, "", 75);
```

За да се прикачи календарът се взема полето в отделна променлива, която се използва в конструктора на календара. Форматът на датата се установява чрез функцията `setDateFormat("%Y-%m-%d")`.

```
input_from = myMenu.getInput("date_from");
myCalendar = new dhtmlXCalendarObject([input_from]);
myCalendar.setDateFormat("%Y-%m-%d");
```

Филтърът стартира при скриване на календара, закачен към полето. Това става чрез прикачване на събитие към календара "onHide". Използва се методът за филтриране на обектът, който визуализира таблицата `myGrid.filterBy(1, function (a)`. Като параметри му се подават номер на колона и функция, по която ще се извърши филтрирането.

```
if (myMenu.getValue("date_from") != "") {
    myGrid.filterBy(1, function (a) {
        return (a == myMenu.getValue("date_from"));
    });
}
```

Останалите два бутона – „премахване“ и „обнови“ извършват операциите си при събитието „onClick“ прикачено за лентата с бутони. Първият вика функцията deleteTask(), която изтрива задачата, по нейния идентификатор. Тъй като след изтриване тя би следвало да липсва в таблицата, се налага презареждане на съдържанието. Функцията updateGrids(); Презарежда на ново всички таблици в потребителската страница. Така съдържанието на таблицата става актуално. При селектиране на ред от таблицата автоматично се попълва формулярът на задачата. Ако тя обаче бъде изтрита, формулярът се зачиства също. Бутонът обнови премахва филтрите, като обновява съдържанието на таблицата.

Другата група бутони от лентата се задействат при събитието „onClick“, прикачено за нея. По идентификатор на бутона се разбира кой от тях е бил избран.

Филтрите са четири:

- Задължителни - показва само задължителните задачи
- Незадължителни- показва незадължителните задачи
- Приключени – показва приключени задачи
- Неприключени- показва неприключени задачи

Във втория раздел на облика е прикачен dhtmlx формуляр. Той служи за визуализация на задачата, селектирана в таблицата.

Формулярът има следните полета:

- Заглавие – текстово поле
- Номер – текстово поле
- Описание – текстово поле
- Дата- текстово поле с календар
- Приключена - отметка
- Задължителна –отметка

Съдържанието на формуляра се променя динамично при събитието "onRowSelect".

```
myGrid.attachEvent("onRowSelect", doOnRowSelected)
```

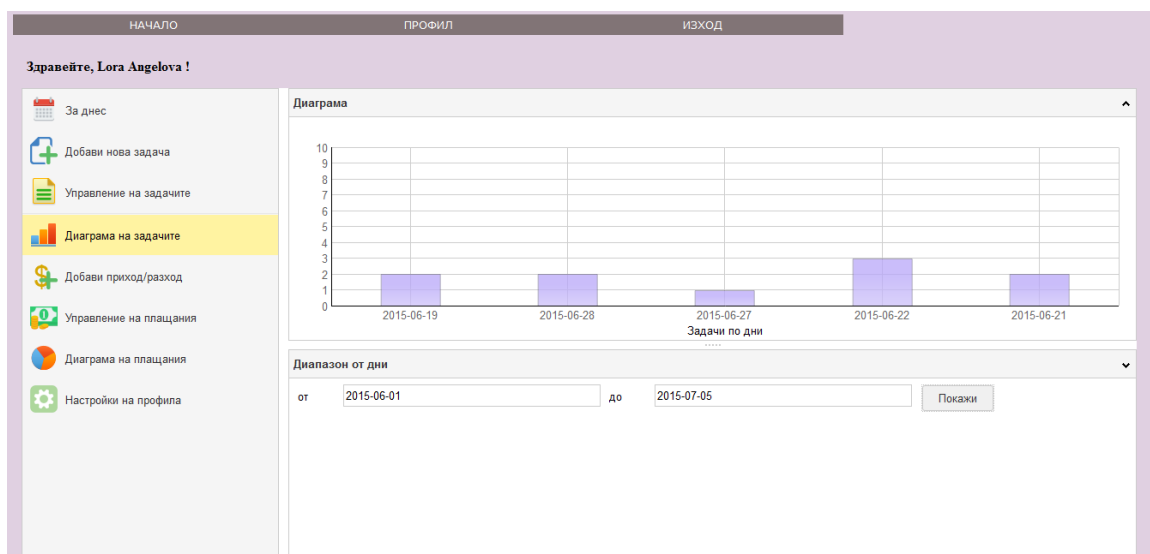
Функцията doOnRowSelected() попълва формуляра с данните от избрания ред в таблицата. Чрез избиране на бутон „запази“ данните се съхраняват в базата чрез заявка за обновяване, като на скрипта със заявката се праща номерът на задачата и всички нейни данни. След съхранения формулярът се зачиства.

```

myFormView.attachEvent("onButtonClick", function (id) {
    if (id == "save") {
        myFormView.send("updateTask.php", "post", function (loader, response) {
            alert(response);
            updateGrids();
            clearMyFormView();
        });
    }
});

```

3.1.4.4 Диаграма на задачите



Тази страница има за цел да даде по-различен изглед на съществуващите задачи. Акцентът е върху дните и броя на съществуващите задачи във всеки от тях. За да се осъществи това, се използва диаграма и формуляр от Java Script библиотеката dhtmlx.

Формулярът и графиката са прикачени към оформление. Графиката е dhtmlx компонент, който служи за визуализация на различни входни данни, под формата на диаграми. Те могат да бъдат няколко вида, с по няколко различни стила.

```

myLayoutTaskStat = mySidebar.cells("day").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Диаграма"}, {id: "b",
text: "Диапазон от дни"}]
});

```

Формулярът съдържа две полета и един бутон, като тези елементи са описани в json файла `formStatTasks.json`. Чрез тези контроли се определя диапазонът на датите, за които да се покажат задачите.

- От- определя начална дата за диапазона
- До- определя крайна дата за диапазона
- Покажи- при натискане на този бутон се визуализират дните, като големината на правоъгълниците съответства на натовареността на деня (броя на задачите с тази дата).

Вземането на данните се осъществява на сървъра, като php скрипт праща заявка до базата данни.

```
myFormStatTasks.attachEvent("onButtonClick", function (id) {  
    if (id == "go") {  
        myFormStatTasks.send("getStatTasks.php",  
            "post", function (loader, response) {  
                alert(response);  
                attachChartTasks();  
            });  
    }  
})
```

След като са налице необходимите данни, се вика функцията `attachChartTasks()`; Тя има за цел да конструира обекта на диаграмата като я прикачи към раздел „a“ на изгледа и да я визуализира. Задават се шаблоните, по които да се вземат данните от xml файла и настройките на изгледа (цвят, рамка). Диаграмата има две измерения. На абцисата са изобразени датите, а на ординатата броя дни- от нула до десет със стъпка едно.

```
function attachChartTasks() {  
myChartTasks = myLayoutTaskStat.cells("a").attachChart({  
    view: "bar",  
    container: "chartDiv",  
    value: "#taskscount#",  
    gradient: "falling",  
    color: "#b9a8f9",  
    radius: 0,  
    alpha: 0.5,  
    border: true,
```

```

        width: 70,
        xAxis: {
            template: "#date#",
            title: "Задачи по дни"
        },
        yAxis: {
            start: 0,
            end: 10,
            step: 1,
        }
    });

    myChartTasks.load("xml/chartTasks.xml");
    myChartTasks.sort({
        by: "#date#",
        dir: "asc",
        as: "string"
    });
}

```

3.1.4.5 Страница за добавяне на приход/разход.

Страницата за добавяне на приходи/разходи служи за добавяне на плащане. Тъй като приходи и разходи са с подобни полета в базата, те са събрани в една таблица, и се разграничават чрез поле тип (type). Формулярът на страницата служи за добавяне и на

двата вида плащания. Той се инициализира от json файл е прикачен към самия страничен раздел. Съдържа следните контроли:

- Приход/разход- радио бутон, указващ вида на плащането
- Описание – текстово поле, в което се описва плащането.
- Дата- текстово поле с календар, който указва датата на плащането
- Стойност в лева- текстово поле за размера на плащането
- Неотложен- приоритет на плащането
- Запази- Този бутон вика съर्वърен скрипт, който изпраща данните към базата.

След запазване на плащането контролите на формуляра се зачистват и всички таблици в приложението се обновяват.

```
myFormInOut = mySidebar.cells("addInOut").attachForm();  
myFormInOut.loadStruct("json/formInOut.json");  
myFormInOut.attachEvent("onButtonClick", function (id) {  
    if (id == "send") {  
        myFormInOut.send("savingInOut.php", "post", function (loader, response)  
        {  
            alert(response);  
            updateGrids();  
        });  
        clearMyFormInOut();  
    }  
});
```

3.1.4.6 Управление на плащания

НАЧАЛО ПРОФИЛ ИЗХОД

Здравейте, Lora Angelova !

За днес

- Добави нова задача
- Управление на задачите
- Диаграма на задачите
- Добави приход/разход
- Управление на плащания**
- Диаграма на плащания
- Настройки на профила

За дата:

Приходи и разходи

НОМЕР	ВИД	ДАТА	ОПИСАНИЕ	РАЗМЕР	ЗАДЪЛЖИТЕЛНА	ПРИКЛЮЧЕНА
9	Приход	2015-06-22	Заплата	1500	Да	Да
14	Приход	2015-06-22	Стипендия	50	Не	Не
15	Разход	2015-06-28	Наем	100	Не	Не
16	Приход	2015-06-20	Пътни	40	Да	Не

Преглед

Преглед на плащане: 15

Приход ☐ Разход ☒

Дата*

Размер*

Описание*

Приключен ☐ Неотложен ☐

Разделът служи за управление на приходи и разходи. Управлението се осъществява с помощта на различни филтри, редакция и изтриване на плащания.

Интерфейсът на страницата е изграден от оформление, прикачено за страничния раздел, таблица, лента с бутони и формуляр.

Лентата има две групи бутони. В първата група се намира филтър за дата. Той представлява текстово поле с календар, прикачено за лентата. Филтрацията се извършва при скриване на календара. В таблицата остават всички плащания за избраната дата.

Операциите за всички бутони се извършват при събитието „onClick“. Бутоните се разпознават по id.

Бутонът „премахни“ служи за изтриване на плащане. При избор на този бутон се намира номера на задачата, чрез селектирания ред и клетката за номер и се подава на php скрипт. Това става чрез изпращане на AJAX post заявка.

```
myMenuPayments.attachEvent("onClick", function (id) {
    if (id == "delete"){
        var selectionPayments = myGridPayments.getSelectedRowId();
        var valueCellPayments = myGridPayments.cells(selectionPayments,
            0).getValue();
        deletePayment(valueCellPayments);
        myGridPayments.deleteSelectedItem();
        updateGrids();
        clearMyFormViewPayments();
    }
});
```

След изтриване на плащане се обновяват всички таблици с функцията `updateGrids()`. Тъй като има дефинирано събитие при селектиране на ред- за попълване на формуляра, то при изтриване, неговите полета също трябва да се зачистват. За тази цел се използва функцията `clearMyFormViewPayments()`;

Бутонът за обновяване вика функцията `updateGrids()`. Останалата група бутони извършват филтриращи операции с dhtmlx функцията на таблицата- `filterBy()`.

```
if (id == "incomes")
    myGridPayments.filterBy(1, function (a) {
        return (a == "Приход");
    });
if (id == "outcomes")
    myGridPayments.filterBy(1, function (a) {
        return (a == "Разход");
    });
```

```

});
if (id == "filterFinished")
    myGridPayments.filterBy(5, function (a) {
        return (a == "Да");
    });
if (id == "filterUnfinished")
    myGridPayments.filterBy(5, function (a) {
        return (a == "Не");
    });
});

```

Таблицата с плащания е прикачена към раздела на облика. Тя се инициализира от XML файл. Събитието при селектиране на ред служи за попълване на формуляра. Така може едно плащане да се редактира.

Формулярът е прикачен към другият раздел на облика. Той се инициализира от json файл и има следните полета:

- Номер-текстово поле с номера на плащането. То не може да се редактира, а се използва само за четене. lock()
- Приход/разход – радио бутон указва типа на плащането
- Дата- текстово поле с календар за датата на плащането
- Описание- многоредово текстово поле с информация за плащането
- Размер- текстово поле -сумата на плащането в лева
- Неотложен- приоритет на плащането
- Приключен- статус на плащането

```

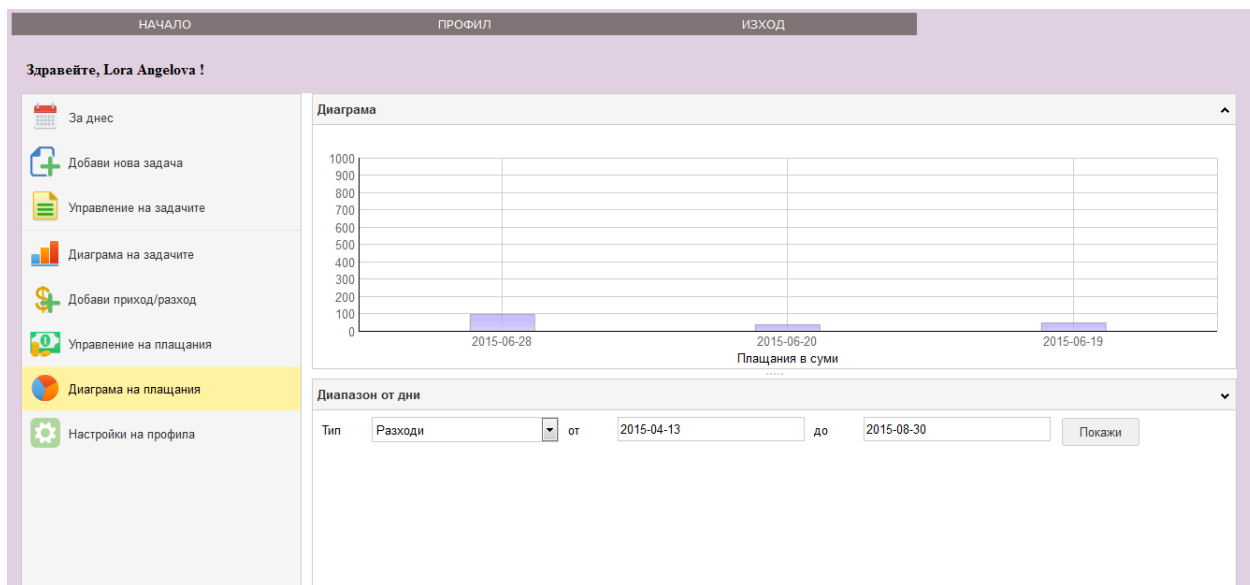
myFormViewPayments = myLayoutPayments.cells("b").attachForm();
myFormViewPayments.loadStruct("json/formViewPayments.json");
myFormViewPayments.attachEvent("onButtonClick", function (id) {
    if (id == "save") {
        myFormViewPayments.send("updatePayments.php", "post", function
            (loader, response) {
                alert(response);
                updateGrids();
                clearMyFormViewPayments();
            });
    }
});

```

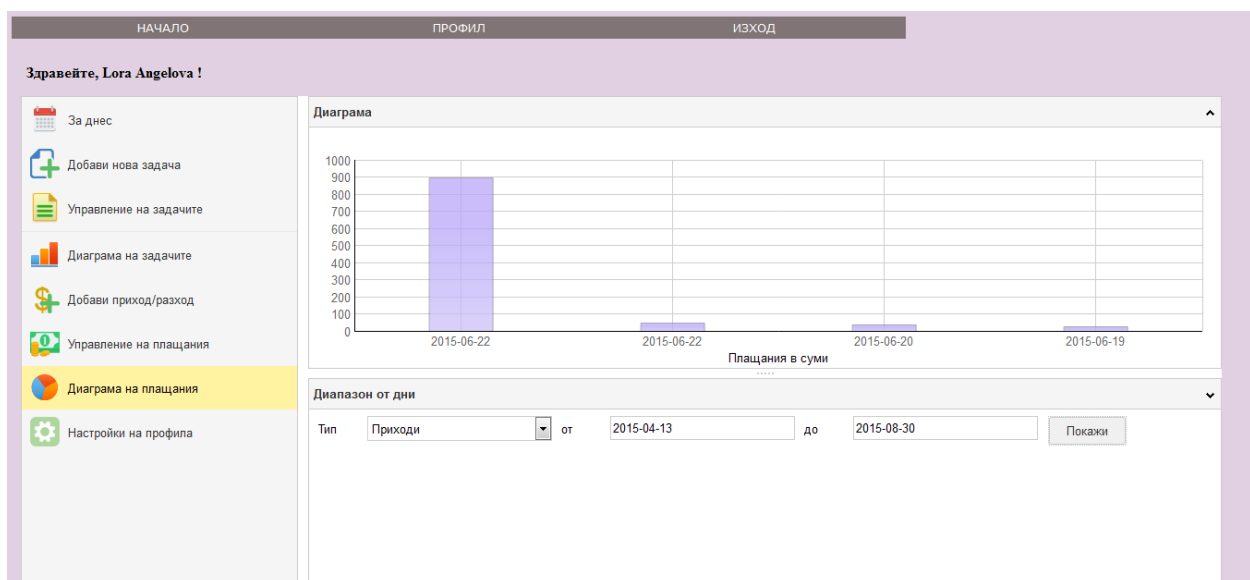
При избор на бутона „запази“, данните от формуляра се изпращат към сървърния скрипт, чрез функцията `send()` на формуляра. След това формулярът се изчиства, и всички таблици в приложението се обновяват.

3.1.4.7 Диаграма на плащанията

Диаграма на приходи



Диаграма на разходи



Разделът с диаграми на плащанията служи за визуализация на различен изглед. Във времеви диапазон се показват приходи или разходи- в зависимост кой тип плащане е избран за визуализация

Интерфейсът се състои от облик с два раздела, като за единия се прикачва диаграмата(при избор на бутона „Покажи“), а в другия се помества формуляра.

Формулярът се инициализира от xml файл и съдържа следните контроли:

- Тип- падащо меню, служи за избор на типа на плащанията, който ще се визуализират- приходи или разходи.
- От- текстово поле с календар за начална дата на диапазона
- До- текстово поле с календар за крайна дата на диапазона
- Покажи- бутон, при чието натискане се инициализира диаграмата в раздел „а “ на облика.

Диаграмата се изчертава с помощта на dhtml компонент –chart. На абсцисата са изобразени датите, а на ординатата стойностите в лева, като започват от нула до хиляда, със стъпка сто. При конструирането и се указва кои данни да използва от XML файла, настройват се стойностите на абсцисата и ординатата- цвят, рамка и сортировки.

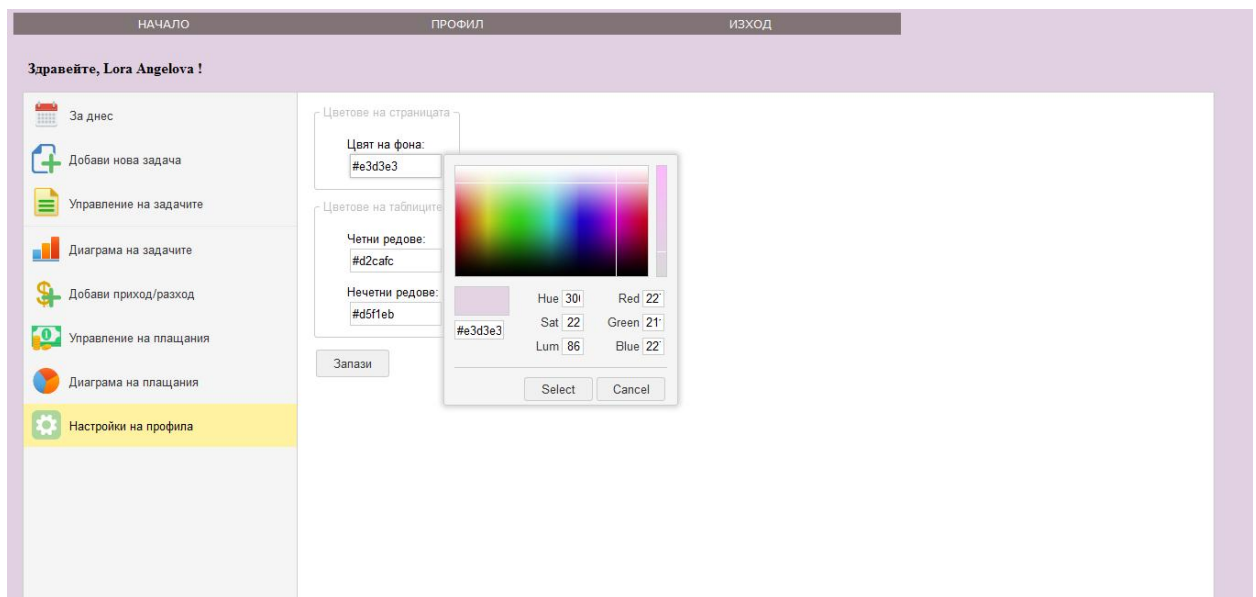
```
function attachChartPayments() {  
    myChartPayments = myLayoutPaymentStat.cells("a").attachChart({  
        view: "bar",  
        container: "chartDiv",  
        value: "#amount#",  
        gradient: "falling",  
        color: "#b9a8f9",  
        radius: 0,  
        alpha: 0.5,  
        border: true,  
        width: 70,  
        xAxis: {  
            template: "#date#",  
            title: "Плащания в суми"  
        },  
        yAxis: {  
            start: 0,  
            end: 1000,  
            step: 100,  
        }  
    });  
    myChartPayments.load("xml/chartPayments.xml");  
    myChartPayments.sort({  
        by: "#date#",
```

```

        dir: "asc",
        as: "string"
    });
}

```

3.1.7.8 Раздел с настройки на профила



Тъй като една от основните идеи на приложението е да се демонстрира работа с JavaScript библиотеки за изграждане на потребителски интерфейс, в този раздел се демонстрира чрез семпъл пример персонализирането на интерфейс, чрез персонални данни съхраняващи се в база от данни. Към разделът е прикачен формуляр, с две групи полета с палитра за цветове, които се инициализират с цветовете константи по подразбиране.

```

{type: "colorpicker", position: "label-left", name: "even", label: "Четни редове:", value: "#d2cafc"},

```

Промяната на интерфейса се осъществява като тези данни се съхранят и страницата се презареди. След съхранение това се извършва автоматично- `location.reload()`;

```

<style>

    .even{
        background-color:<?php echo $_SESSION['even'];?> ;
    }

```

```

        .uneven{
            background-color:<?php echo $_SESSION['uneven']; ?>;
        }
    </style>

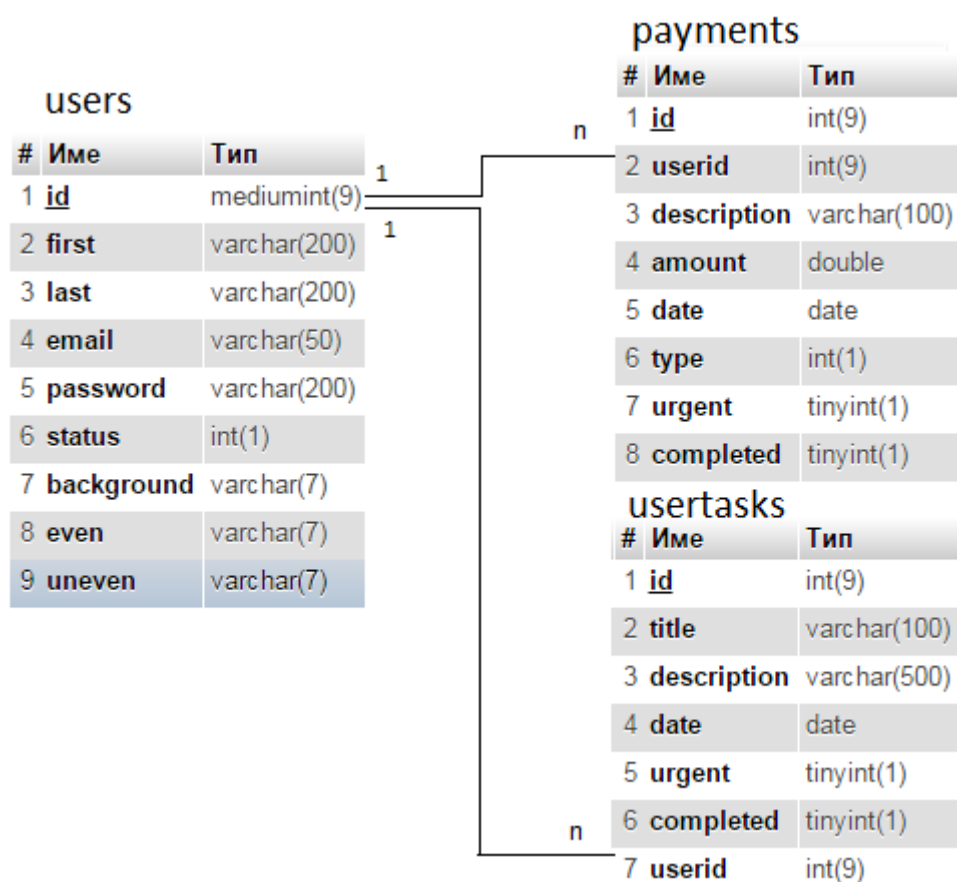
```

Функцията `enableAlterCss()` позволява да се приложи стил на таблицата, а самия стил се формира динамично с помощта на изхода от сървърен скрипт, който представлява разпечатка на данните за цветовете.

```
myGridTodayTasks.enableAlterCss("even", "uneven");
```

3.2 Описание на сървърната част на клиентското приложение, описание на функциите и употребата на технологиите. Обосновка

3.2.1 Структура на база данни



Базата данни се състои от три таблици:

- users
- usertasks
- payments

Връзките в таблицата се осъществяват по външен ключ –id на потребител, в таблиците usertasks и payments. Един потребител може да има много задачи и плащания, а една задача или едно плащане-само един потребител.

Users

Таблицата служи за съхранение на данни за потребител. Попълва се при регистрация и се използва при вход , като от нея се вземат данните на потребителя, чрез който се осъществяват релациите с другите таблици и се визуализира страницата.

Поleta:

- id –поле от тип integer с размер 9 знака. Служи за съхранение на потребителски идентификационен номер. Полето е първичен ключ.
- first- поле тип VARCHAR с размер 200 символа - Съхранява първото име на потребителя.
- last- поле тип VARCHAR с размер 200 символа - Съхранява фамилното име на потребителя.
- email – поле тип VARCHAR с размер 50 символа- Съхранява електронния адрес на потребителя
- password- поле тип VARCHAR с размер 200 символа- Съхранява паролата на потребителя
- background- поле тип VARCHAR с размер 7 символа- Съхранява цвятова константа за фона на потребителската страница
- even- поле тип VARCHAR с размер 7 символа- Съхранява цвятова константа за четните редове в таблиците на потребителската страница
- uneven - поле тип VARCHAR с размер 7 символа- Съхранява цвятова константа за нечетните редове в таблиците на потребителската страница

usertasks

Таблицата служи за съхранение на задачи за всички потребители. Използва се добавяне , редакция и изтриване на задача.

Поleta:

- id- поле тип integer с размер 9. Съхранява идентичен номер на задача. Полето е първичен ключ.
- title- поле тип VARCHAR с размер 100 символа- Съхранява заглавието на задача.
- description- поле тип VARCHAR с размер 500 символа- Съхранява описанието на задача
- date -поле тип date. Съхранява датата, за която е запланувана задачата
- urgent- поле тип boolean. Съхранява приоритета на задача
- completed- поле тип boolean. Съхранява статуса на задача
- userid- поле тип integer с размер 9 символа- Съхранява id на потребителяПолето е външен ключ.

payments

В тази таблица се съхраняват плащанията на всички потребители. Плащанията са два вида- приходи и разходи. Видът на плащането се определя от колоната type- 0-разход, 1- приход. Таблицата се използва за добавяне редактиране и изтриване на плащане.

Полета:

- id- поле тип integer с размер 9. Съхранява идентичен номер на плащане. Полето е първичен ключ.
- userid- поле тип integer с размер 9 символа- Съхранява id на потребителя. Полето се явява външен ключ.
- description- поле тип VARCHAR с размер 100 символа- Съхранява описанието на плащане.
- amount- поле тип double. Съхранява стойността на плащането.
- date- поле тип date. Съхранява датата на плащането.
- type-поле от тип boolean. Съхранява типа на плащането.
- urgent- поле тип boolean. Съхранява приоритета на плащането.
- completed- поле тип boolean. Съхранява статуса на плащането.

3.2.2 Връзка с база данни

За връзката с базата данни се използва скрипта setup.php. Тя представлява скрипт с конекция .

```
$dbc = mysqli_connect('localhost', 'dev', 'loapsw', 'budget_planning') OR  
die('Error' . mysqli_connect_errno());
```

На функцията се подава адреса, името на потребителя, паролата и името на базата .
Той се включва, в останалите скриптове, който извършват операции в базата с данни.
`include('setup.php');`

3.2.3 Регистрация

registerme.php

Създаването на регистрацията представлява добавяне на нов ред в таблицата за потребители. По този начин при вход, системата ще открие записът, ще провери паролата и ще позволи достъп до потребителската страница.

За да се добави запис, първо трябва да се провери дали вече съществува потребител със същото потребителско име. Затова първоначално се прави заявка и се извличат съдържанието на таблицата users.

```
$q = "SELECT email FROM users ";
```

Всички електронни адреси (потребителски имена) се съхраняват в масив. Ако потребителското име, което е било изпратено от клиентския скрипт не съществува в масива с имена, се прави заявка към базата с данни, като и се подават останалите стойности от POST масива и подразбиращите се цветови настройки.

```
$qINSERT = "INSERT INTO users (first,last,email,password,background,even,  
uneven) VALUES  
( '$first', '$last', '$email', '$psw', '#e3d3e3', '#d2cafc', '#d5f1eb' );";
```

Ако потребителското име съществува се извежда съобщение за грешка.

3.2.4 Вход в системата

За входа в системата отговаря скриптът login.php. В него се включва конекцията с базата и след това се прави заявка, като за параметри и се подават данни от POST масива.

```
$q = "SELECT * FROM users where email='$_POST[user]' AND  
password='$_POST[password]';
```

Ако има открит ред с тези данни, тогава се установяват потребителските данни в сесийни променливи и потребителят се пренасочва към потребителската страница.

Данни за таблиците в приложението

В приложението съществуват два вида записи:

- плащания
- задачи

За управлението им се използват четири таблици- по две за всеки вид.

- Задачи за днешен ден- getTodayTasks.php
- Плащания за днешен ден -getTodayPayments.php
- Всички задачи -getTasks.php
- Всички плащания - getPayments.php

Всички таблици използват при инициализацията си xml файлове. Тези файлове се създават с помощта на php скриптове, които вземат данни от базата чрез заявки и след това сформират xml структурите.

И четирите скрипта са на един и същ принцип. Първоначално се прави заявка към базата. Колоните на таблиците- еднократно.

```
$userid=$_SESSION['userid'];  
include('setup.php');  
  
$q = "SELECT * FROM usertasks WHERE userid='$userid';  
$r = mysqli_query($dbc, $q);  
  
$xml="<rows> \n \t";  
$xml.="<head> \n \t\t" ;  
$xml.="<column width=\"70\" type=\"ro\" align=\"left\" sort=\"str\"> Номер  
</column> \n \t\t";  
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Дата  
</column> \n \t\t";  
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\">  
Заглавие </column> \n \t";  
$xml.="<column width=\"400\" type=\"ro\" align=\"left\" sort=\"str\">  
Описание </column> \n \t";  
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\">  
Задължителна </column> \n \t";
```

```
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\">
Приключена </column> \n \t";
$xml.="</head>\n";
```

След това с помощта на цикъл се сформира низ с данните за всички записи, форматирани според изискванията за xml структурата. Всеки ред има свой номер и няколко клетки, в които се поместват данните от резултата на заявката.

```
$num=1;
while($page = mysqli_fetch_assoc($r))
{
    if($page['urgent']==1) $urgent="Да"; else $urgent="Не";
    if($page['completed']==1) $completed="Да"; else $completed="Не";
    if($page['date']=="0000-00-00") $date="Безсрочна";
    else $date=$page['date'];
    $xml .=" \t <row id=\"".$num.">\n \t\t";
    $xml .=" <cell>".$page['id']."</cell>\n\t\t";
    $xml .=" <cell>".$date."</cell>\n\t\t";
    $xml .=" <cell>".$page['title']."</cell>\n\t\t";
    $xml .=" <cell>".$page['description']."</cell>\n\t";
    $xml .=" <cell>".$urgent."</cell>\n\t";
    $xml .=" <cell>".$completed."</cell>\n\t";
    $xml.="</row>\n\t";
    $num++;
}
```

Накрая се създава XML файлът.

```
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/text.xml");
```

3.2.5 Добавяне на записи

Има два случая в които се добавя запис:

- При нова задача- saveTask.php
- При ново плащане- savePayment.php

Първоначално се включва скриптът за конекция към базата с данни. Данните от пост масива се съхраняват в локални променливи и се прави заявка Insert към базата.

```
$title = $_POST['title'];
$desc = $_POST['description'];
$date=$_POST['date'];
$urgent=$_POST['urgent'];
$userid=$_SESSION['userid'];

$qINSERT = "INSERT INTO usertasks (title,description,date,urgent,userid)
VALUES ('$title','$desc','$date','$urgent','$userid');";
```

При успех добавянето се потвърждава с php функцията `mysqli_commit()` и след това връзката към базата се затваря `mysqli_close($dbc);`

```
if (mysqli_query($dbc, $qINSERT)) {

    if (!mysqli_commit($dbc)) {
        print("Transaction commit failed\n");
        exit();
    }

    echo "Задачата беше добавена успешно!";
} else {
    echo "Error: " . $qINSERT . "<br>".
        mysqli_error($dbc);
}
mysqli_close($dbc);
include('updateXMLs.php');
```

Тъй като информацията за визуализиране при инициализация на страницата вече не е актуална, се вика скрипт, който от своя стана вика наново скриптовете, които вземат актуалните данни и създават xml файловете- `updateXMLs.php`

updateXMLs.php

```
include('getTasks.php');
include('getTodayTasks.php');
include('getPayments.php');
include('getTodayPayments.php');
```

3.2.6 Обновяване на записи

Обновяването на записи представлява заявка update към базата данни. Могат да се променят всички колони или само някои от тях. В приложението се използва за установяване на плащане и задача като завършени- в раздела за днешна дата и за цялостна редакция на данните- при управление на задачи и плащания.

Обновяване на колона от записа се извършва чрез скриптовите SetPaymentCompleted.php и setTaskCompleted.php.

Осъществява се конекция към базата, чрез включване на скрипта setup.php. На скриптовите за обновяване се подава id чрез POST заявка.

```
$taskid=$_POST['taskId'];  
$qUPDATE = "UPDATE usertasks SET completed=true WHERE id='$taskid';";
```

След приключване на операцията конекцията към базата се затваря и се обновяват всички таблици в приложението с новите данни.

Обновяване на цялата информация

Три са скриптовите, които извършват цялостно обновяване:

- updateTask.php-обновява данните по запис на задача
- updatePayment.php- обновява данните по запис на плащане
- updateSettings.php- обновява потребителските настройки за цветове

Цялостното обновяване се извършва по аналогичен начин. На php скрипта се подават всички данни за записа чрез POST заявка. След това се сформира sql израз.

```
$qUPDATE = "UPDATE usertasks SET  
title='$title',description='$desc',date='$date',urgent='$urgent',completed='$  
completed' WHERE id='$taskid';";
```

Заявката се изпраща `mysqli_query($dbc, $qUPDATE)` и след нейното изпълнение се потвърждават промените. Таблиците в приложението също се обновяват с новите данни.

3.2.7 Изтриване на записи

Има два случая, където се използва изтриване на запис:

- Изтриване на задача- deleteTask.php
- Изтриване на плащане- deletePayment.php

Изтриването става по id на записа. То се подава чрез post заявка на скриптът, който го извършва.

```
$qDELETE = "DELETE FROM usertasks WHERE id='$id';";
```

Отново се потвърждава промяна в базата и се обновяват таблиците в приложението.

Потребителски настройки

Потребителските настройки се състоят в три полета от потребителската таблица, които указват цвят на фон, четен и нечетен ред на таблицата. Тези данни се взимат при стартиране на приложението и чрез тях се сформира потребителския интерфейс. Това става с обикновена select заявка. След това резултатите се съхраняват в променливи, а те се извеждат от сървърните скриптове на местата където са необходими цветовете константи. Така динамично се променя облика на приложението.

session_start();

```
include('setup.php');
include('globals.php');
$userid=$_SESSION['userid'];

$qSettings = "SELECT * FROM usersettings where userid='$userid'";
$rSettings = mysqli_query($dbc, $qSettings);
$userdataSettings= mysqli_fetch_assoc($rSettings);

$_SESSION['background'] = $userdataSettings['background'];
$_SESSION['even']=$userdataSettings['even'];
$_SESSION['uneven']=$userdataSettings['uneven'];
```

3.2.8 Изход

logout.php

Достъпът до потребителската страница се осъществява само ако е установена променливата `$_SESSION['username']`. В противен случай потребителят се препраща към страницата за вход. След вход, тази променлива се установява с някаква стойност.

```
if (!isset($_SESSION['username'])) {  
    header('Location: login.php');  
}
```

Изходът от приложението се осъществява в скрипта `logout.php`

Изходът има за цел да разруши всички сесийни променливи и пренасочи потребителя към страницата за вход. Така потребителят не може да използва потребителската страница, защото при повторен опит за достъп тя ще види, че променливата не е установена и няма да го допусне.

```
session_start();  
#Изтриване на username  
unset($_SESSION['username']);  
#Изтрива всички ключове  
session_destroy();  
header('Location: login.php');  
?>
```

4. Документации

4.1 Използвани функции на Java Script библиотеката

4.1.1 *Формуляр*

`dhtmlxForm` е JavaScript компонент, чрез който се генерират и упеавляват Ajax-базирани уеб формуляри със стандартно множество от елементи като: текстово поле, радио бутони, падащи менюта и т.н. Компонентите на формуляра предлагат контрол върху качвания на файлове и валидации, и възможност за съхранение в база данни.

Използван начин на инициализация

JSON формат- най препоръчваният формат, тъй като е най-лесен за управление.

JSON файл

```
[
  {type:"fieldset", name:"data", label:"Welcome", inputWidth:"auto", list:[
    {type:"input",    name: "name", label:"Login"},
    {type:"password", name:"pass",  label:"Password"},
    {type:"button",   name:"save",  value:"Proceed"}}
  ]
}
```

HTML код:

welcome(json).html

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
  href="codebase/skins/dhtmlxform_dhx_skyblue.css">
  <script src="codebase/dhtmlx.js" type="text/javascript"></script>
  <link rel="STYLESHEET" type="text/css" href="codebase/dhtmlx.css">
</head>

<body onload="doOnLoad()">
  <div id="form_container" style="width:250px;height:300px;"></div>
  <script>
    var myForm, formData;
    function doOnLoad() {
      formData = [
        {type:"fieldset", name:"data", label:"Welcome",
          inputWidth:"auto",
          list:[
            {type:"input",    name:'name', label:'Login'},
            {type:"password", name:"pass",  label:"Password"},
            {type:"button",   name:"save",  value:"Proceed"}}
          ]
      }
      myForm = new dhtmlXForm("form_container", formData);
    }
  </script>

</body>
```

```
</html>
```

Използвани контроли:

- Button -стандартен бутон
- Calendar -поле, което при селектиране отваря изкачащ прозорец с календар.
- Checkbox -поле за отметка
- Colorpicker -поле, което при селектиране отваря изкачащ прозорец за избор на цвят.
- Fieldset -обединител за смислено-свързани полета
- Input/Textarea -поле за въвеждане на текст
- Label -текстов етикет
- Newcolumn -позволява създаването на нова колона с контроли
- Password -текстово поле със скрити символи.
- Radio -стандартен радио-бутон.
- Settings -позволява предефиниция на настройките за компонентите на формуляра

Използвани методи:

attachEvent -добавя дефинирано от потребителя действие, при настъпването на някакво събитие

```
number attachEvent(string name,function handler);
```

Параметри

- name string име на събитието
- handler function потребителски дефинирана функция

Връща

- number internal event id може да се използва за detachEvent(id)

```
myFormView.attachEvent("onButtonClick", function (id) {  
    ...  
})
```

disableItem -установява елементът да е неактивен

```
void disableItem(string name,mixed value);
```

Параметри:

- name string името на елемента
- value mixed стойността на елемента (за радио бутон)

```
myFormViewPayments.disableItem("id");
```

getInput -връща обекта на съответния елемент (input, textarea, password, calendar, colorpicker, hidden only)

object getInput(string name);

Параметри

- name string- име на елемент

Връща

- Обект на елемент. Ако такъв не е намерен връща null.

```
input_from1 = myMenuPayments.getInput("date_from");
```

loadStruct -зарежда данни в компонента чрез xml или json

void loadStruct(mixed data,function doOnLoad);

Параметри:

- data смесен xml/json url/string/object
- doOnLoad функция (optional) извиква потребителски дефинирана функция след като бъде заредена структурата

```
myFormViewPayments.loadStruct("json/formViewPayments.json");
```

send -изпраща данните на сървърен скрипт

void send(string url,bool mode,function callback);

Параметри:

- url string сървърният скрипт, който ще приеме данните
- mode bool режимът на изпращане- може да бъде get или post. По подразбиране е post.
- callback function методът, който да се извика след изпращане. (опционално)

```
myFormViewPayments.send("updatePayments.php", "post", function (loader,
response) {
    alert(response);
    updateGrids();
});
```

```
clearMyFormViewPayments();  
});
```

setFontSize -установява размера на шрифта на целия формуляр

```
void setFontSize(mixed size);
```

Параметри:

- size mixed размер на шрифта във формуляра

```
myForm.setFontSize("15px");
```

setItemValue- -установява стойността на елемент

```
void setItemValue(string name,mixed value);
```

Параметри:

- name string името на елемента
- value mixed стойност за елемента

```
myForm.setItemValue("id", "");
```

validate- -стартира валидацията на формуляра. Използва правилата , дефинирани при структурата на формуляра.

```
void validate();
```

```
myForm.validate()
```

4.1.2 Страничен раздел

dhtmlxSidebar компонента представлява вертикално меню за навигация. Към него могат да се добавят точки, а към тях да се прикачват други DHTMLX компоненти. Има пет предефинирани шаблона (details, tiles, icons, icons_text, text) за визуализация на елементите.

Използван начин на инициализация:

json формат

```
{items: [
```

```
{id: "today",text: "За днес", icon: "today.png", selected: true},
{id: "addTask", text: "Добави нова задача",      icon: "add.png"},
{id: "manageTasks",text: "Управление на задачите", icon:"documents.png"},
{type: "separator"},
{id: "day", text: "Диаграма на задачите",  icon: "chart.png"},
{id: "addInOut", text: "Добави приход/разход", icon: "addMoney2.png"},
{id: "managePayments", text: "Управление на плащания", icon: "money.png"},
{id: "calculate", text: "Диаграма на плащания",  icon: "pie.png"},
{id: "settings", text: "Настройки на профила",  icon: "settings.png"},
]}
```

```
function doOnLoad() {

    mySidebar = new dhtmlXSideBar({
        parent: "sidebarObj",
        icons_path: "common/win_32x32/",
        template: "tiles",
        width: 300,
        json: "json/sidebar.json",

        ...
    })
}
```

Използвани методи:

attachLayout- Прикачва оформление, в клетката на страничния раздел

dhtmlXLayoutObject attachLayout(mixed conf);

Параметри

- conf mixed layout pattern или api-init config

Връща

- dhtmlXLayoutObject dhtmlXLayoutObject инстанция

```
myLayoutPaymentStat = mySidebar.cells("calculate").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Диаграма"}, {id: "b",
text: "Диапазон от дни"}]
});
```

attachForm –добавя формуляр към клетка от страничния раздел

dhtmlXForm attachForm(object conf);

Параметри:

- `conf` конфигурация на обект формуляр

Връща

- `dhtmlXForm` `dhtmlXForm` инстанция

```
myFormSettingsB = mySidebar.cells("settings").attachForm();
```

attachToolbar- добавя лента с бутони към клетката от раздела

`dhtmlXToolbarObject attachToolbar(object conf);`

Параметри:

- `conf` `object`

Връща

- `dhtmlXToolbarObject` `dhtmlXToolbarObject` инстанция

```
myMenuTodayTasks = myLayoutToday.cells("a").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarToday.xml"
});
```

4.1.3 Оформление

С помощта на `dhtmlxLayout` компонента става възможна комбинацията на няколко `dhtmlx` компонента на една страница, които могат да имат съвсем различно съдържание.

Използвани методи:

attachChart- добавя диаграма към клетка

`dhtmlxChart attachChart(object conf);`

Параметри:

- `conf` `object`- конфигурация на диаграма

Връща:

- `dhtmlXChart`- `dhtmlXChart` инстанция

```
myChartTasks = myLayoutTaskStat.cells("a").attachChart({
    ...
});
```

attachForm- прикачва `dhtmlxForm` към клетка.

`dhtmlXForm attachForm(object conf);`

Параметри:

- `conf` конфигурация на формуляр.

Връща:

- `dhtmlXForm` `dhtmlXForm` инстанция

```
myForm = mySidebar.cells("addTask").attachForm();
```

attachToolbar прикачва `dhtmlXToolbar` към клетка

`dhtmlXToolbarObject attachToolbar(object conf);`

Параметри:

- `conf` `object`

Връща:

- `dhtmlXToolbarObject` `dhtmlXToolbarObject` инстанция

```
myMenuTodayPayments = myLayoutToday.cells("b").attachToolbar({  
    ...  
})
```

4.1.4 Таблици

`dhtmlXGrid` се използва за визуализация на таблици, с възможност да се извършват операции над тях като филтриране и редактиране на клетки.

Използван метод за инициализация:

Всички таблици в приложението се инициализират с помощта на `xml` файл.

```
myGridTodayTasks.loadXML("xml/todayTasks.xml");
```

Използвани методи:

attachEvent-добавя потребителски-дефинирани действия към някое от дефинираните събития

`void attachEvent(string evName,function evHandler);`

Парамтри:

- `evName` `string` име на събитие
- `evHandler` функция на потребителя

```
myGridTodayPayments.attachEvent("onRowSelect", doOnRowSelectedTodayPayments);
```

`enableAlterCss` –установява CSS стил за редовете на таблицата

```
void enableAlterCss(string cssE,string cssU,boolean perLevel,boolean levelUnique);
```

Параметри:

- `cssE` `string`- име на `css` клас за четни редове
- `cssU` `string`- име на `css` клас за нечетни редове
- `perLevel` `boolean` `true/false` -маркира редовете не по ред, а по ниво в дървовидна таблица
- `levelUnique` `boolean` `true/false` – създава допълнителен, уникален `css` клас, базиращ се на нивото на реда в дървовидната таблица

```
.even{
    background-color:#22FF44;
}
.uneven{
    background-color:#41964e;
}

myGridTodayTasks.enableAlterCss("even", "uneven");
```

Използвани събития:

onRowSelect- при избор на ред

```
myGridTodayPayments.attachEvent("onRowSelect", doOnRowSelectedTodayPayments);
```

Лента с бутони

`dhtmlxToolbar` компонента предлага на функционалност за групиране на команди в лента с бутони.

Използван начин на инициализация:

Чрез `xml` файл.

```
<toolbar>
    <item id="delete" type="button"    img="close.gif" imgdis="open_dis.gif"
    text="Премахване"    />
```



```

<item id="reload" type="button"    img="reload.gif"    imgdis="save_dis.gif"
text="Обнови"    />
<item id="sep1"    type="separator"/>
<item id="filterUrgent"    type="button"    img="urgent.png"
imgdis="cut_dis.gif"    text="Задължителни"    />
<item id="filterOptional"    type="button"    img="optional.png"
imgdis="copy_dis.gif"    text="Незадължителни"    />
<item id="filterFinished"    type="button"    img="ok.gif"
imgdis="copy_dis.gif"    text="Приключени"    />
<item id="filterUnfinished"    type="button"    img="list.png"
imgdis="copy_dis.gif"    text="Неприключени"    />
</toolbar>

```

```

myMenu = mySidebar.cells("manageTasks").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarUserPage.xml"
});

```

Използвани методи:

attachEvent- добавя потребителски дефинирано действие към някое от дефинираните събития.

number attachEvent(string name,function handler);

Параметри:

- name string име на събитие
- handler function потребителски дефинирана операция

Връща:

- number id на вътрешно събитие, може да се използва за detachEvent(id)

```

var evId = myComponent.attachEvent("eventName", function(){
    ...
});

```

setItemText- установява текста на елемента

void setItemText(mixed itemId,string text);

Параметри:

- itemId mixed id на елемент
- text string новият текст на елемента

```
myToolbar.setItemText(itemId, text);
```

Използвани събития:

onClick-при избор с мишка

4.1.5 Диаграми

DhtmlxChart е инструмент, с който могат да се създават диаграми за различни уеб приложения. Предлагат се четири вида диаграми, като всеки от тях може да се променя според целта на задачата. Една и съща диаграма може да се представя в двуизмерен и триизмерен вид.

Създаване на диаграма

Има няколко начина за инициализация на диаграма. В проекта се използва инициализация чрез XML файл.

```
<data>
  <item id="1">
    <taskscount>1</taskscount>
    <date>2015-06-19</date>
  </item>
  <item id="2">
    <taskscount>2</taskscount>
    <date>2015-06-28</date>
  </item>
  <item id="3">
    <taskscount>3</taskscount>
    <date>2015-06-22</date>
  </item>
</data>
```

В структурата на диаграмтата се указват стойностите абцисата и ординатата, цвета на правоъгълниците, ширина и рамка.

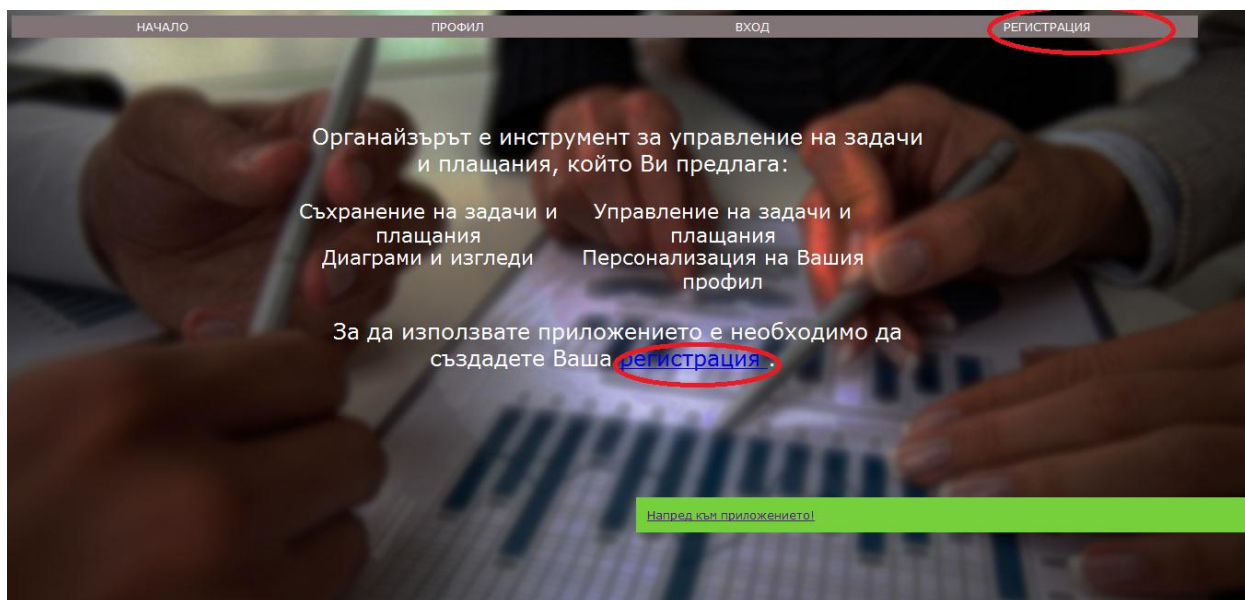
```
myChartTasks = myLayoutTaskStat.cells("a").attachChart({
view: "bar",
  container: "chartDiv",
  value: "#taskscount#",
  gradient: "falling",
  color: "#b9a8f9",
  border: true,
  width: 70,
  xAxis: {
    template: "#date#",
    title: "Задачи по дни"
  },
  yAxis: {
    start: 0,
    end: 10,
    step: 1,
  }
})
```

4.2 Ръководство за потребителя

Органайзърът е инструмент за управление на задачи и плащания, който Ви предлага:

- Съхранение на задачи и плащания
- Управление на задачи и плащания
- Диаграми и изгледи
- Персонализация на Вашия профил

За да използвате приложението е необходимо да създадете Ваша регистрация .



На началната страница на органайзъра се намират два линка, които водят към страницата за регистрация.

След отваряне на страницата се визуализира регистрационната форма. За да създадете Ваш профил трябва да попълните всички полета с коректни данни.

- 1-Вашето име
- 2- Вашата фамилия
- 3- Вашия електронен адрес
- 4-Парола
- 5-Повторете паролата

След приключване на попълването потвърдете Ващите данни, като наиснете бутон „Добави“ (6). В случай, че не успеете да се регистрирате прегледайте наново Ващите данни, коректността на електронния адрес и въведете отново паролите.

Към страницата за вход може да преминете чрез линка в менюто или, този в долния десен ъгъл.

Влезте в профила си, за да използвате приложението!

Електронен адрес :
kdobrev@ebv.bg 1

Парола:
..... 2

Вход 3

На страницата за вход ще намерите формуляр с две полета.

- 1-Въведете Вашия електронен адрес
- 2-Въведете Вашата парола
- 3-Потвърдете данните, като натиснете бутонът „Вход“

Ако се въвели правилно данните си ще бъдете пренасочени към потребителската си страница.

Това е Вашето първо посещение на тази страница и все още нямате добавени задачи или плащания.

За да добавите нова задача отидете в раздела „Добави нова задача“.

НАЧАЛО ПРОФИЛ ИЗХОД

Здравейте, Красен Добрев !

За днес

- Добави нова задача
- Управление на задачите
- Диаграма на задачите
- Добави приход/разход
- Управление на плащания
- Диаграма на плащания
- Настройки на профила

Нова задача

Заглавие * 1

Описание 2

Дата* 3

Неотложна 5 ☐

Добави 6

Изчисти

За да добавите нова задача попълнете данните за нея:

- 1- Заглавието на задачата- попълва се задължително
- 2- Описание- попълва се по Ваш избор
- 3- Дата на задачата
- 4- Приоритета на задачата. По него след това ще управлявате изгледите.

След като попълните данните добавете задачата, като изберете бутон „Натисни“. Ако искате да изчистите данните-бутон „Изчисти“.

След като добавите задачи, може да ги редактирате, изтривате и филтрирате. За целта изберете раздел „Управление на задачите“.

Здравейте, Красен Добрев !

За дата:

НОМЕР	ДАТА	ЗАГЛАВИЕ	ОПИСАНИЕ	ЗАДЪЛЖИТЕЛНА	ПРИКЛЮЧЕНА
150	2015-06-19	Плащания	Да се направи диаграма за плащания.	Да	Да
151	2015-06-28	Настройки на	Да се направи формуляр за настройки.	Да	Да
165	2015-06-22	Home Page	Да се създаде регистрация	Да	Не
166	2015-06-21	Alert	Всички алерти да се преименуват!	Да	Не

Преглед

Нова задача:

Заглавие *: Дата:

Описание:

Неотложна: ☒ Завършена: ☒

За да видите задачите за определена дата изберете дата от календара в лентата.

За дата:

Задачи

НОМЕР	ДАТА	ЗАГЛАВИЕ	ОПИСАНИЕ
165	2015-06-22	Home Page	Да се създаде регистра...
167	2015-06-21	Alert	Всички алерти да се преименуват!
169	2015-06-21	Alert	Всички алерти да се преименуват!

Преглед

Нова задача:

Заглавие *:

За изтриване на дадена задача - кликнете върху нея в таблицата с задачи и изберете бутон „Премахни.“

Ако желаете да редактирате някоя задача, може да го направите като изберете нейния ред в таблицата. Задачата ще се визуализира във формуляра в долната клетка. Можете свободно да промените нейните данни. За да съхраните промените изберете бутон „Запази“.

Здравейте, Красен Добрев !

За дата: 2015-06-22 [Премахване] [Обнови] [Задължителни] [Незадължителни] [Приключени] [Неприключени]

Задачи

НОМЕР	ДАТА	ЗАГЛАВИЕ	ОПИСАНИЕ	ЗАДЪЛЖИТЕЛНА	ПРИКЛЮЧЕНА
150	2015-06-19	Плащания	Да се направи диаграма за плащания.	Да	Да
151	2015-06-28	Настройки на	Да се направи формуляр за настройки.	Да	Да
165	2015-06-22	Home Page	Да се създаде регистрация	Да	Не
166	2015-06-21	Alert	Всички апелти за се приемат	Да	Не

Преглед

Нова задача: 151

Заглавие: Настройки на профила

Дата: 2015-06-28

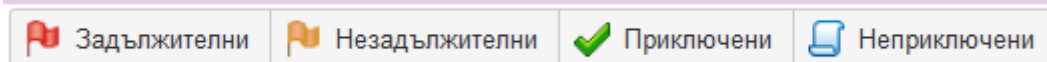
Описание: Да се направи формуляр за настройки.

Неотложна: ☒

Завършена: ☒

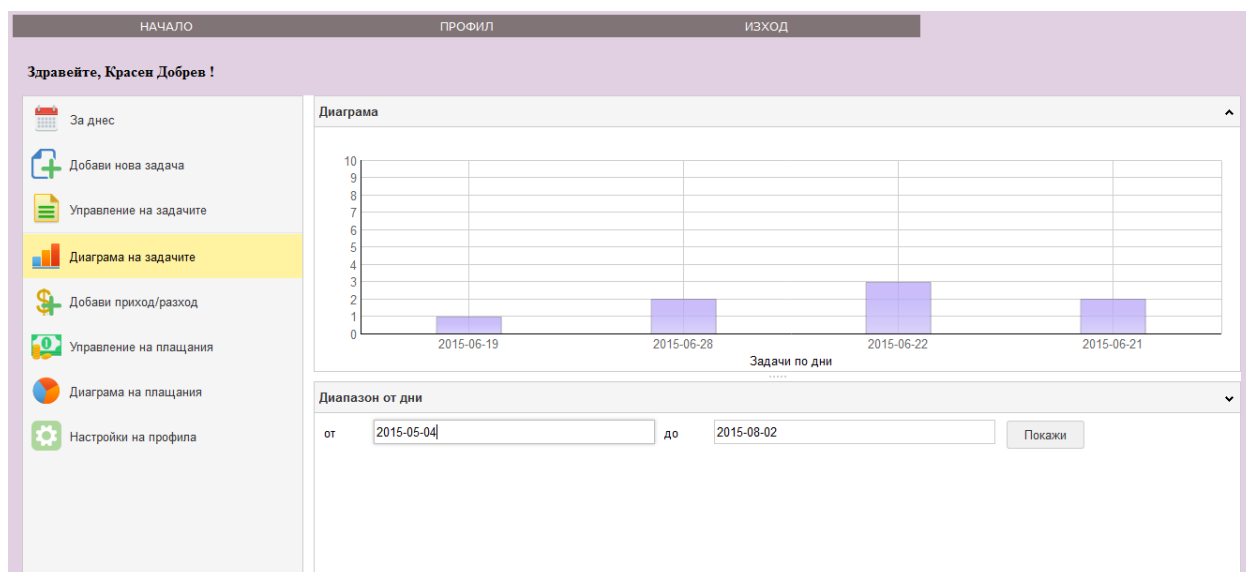
Запази

За да филтрирате Вашите задачи по статус и приоритет използвайте останалите четири бутона.



Ако искате да върнете първоначалния изглед с всички задачи изберете бутон „Обнови“.

За да видите интензивността на задачите по дни, изберете раздела „Диаграма на задачите“.



Вие определяте диапазона от дните, които да се покажат. Изберете начална и крайна дата от формуляра в долния раздел и натиснете бутон „Покажи“, за да се визуализира диаграмата.

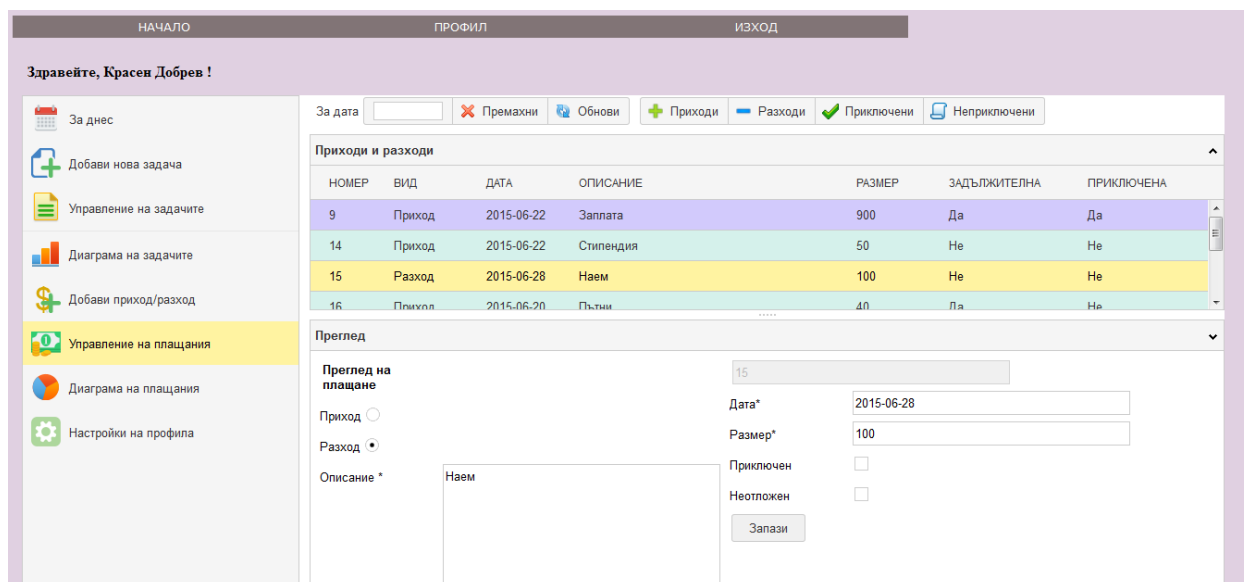
Освен задачи можете да добавяте и плащания. Те са два вида- приходи и разходи. И двата вида се добавят от едно и също място- раздела „Добави приход/разход“.

За да добавите плащане, попълнете следните данни:

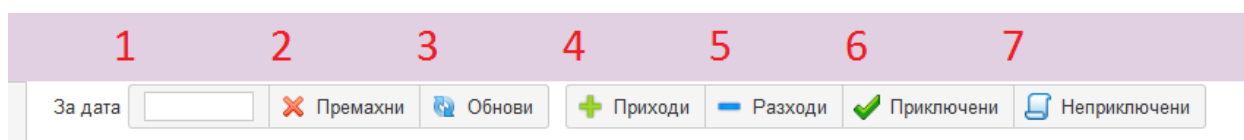
- 1- Изберете вида на плащането- приход или разход
- 2- Добавете описание на плащането
- 3- Изберете дата на плащането
- 4- Въведете сумата на плащането
- 5-Ако плащането е неотложно отбележете това с отметка.

След въвеждане на необходимите данни изберете бутон „Запази“.

Както при задачите, така и при плащанията са възможни операции върху записите.



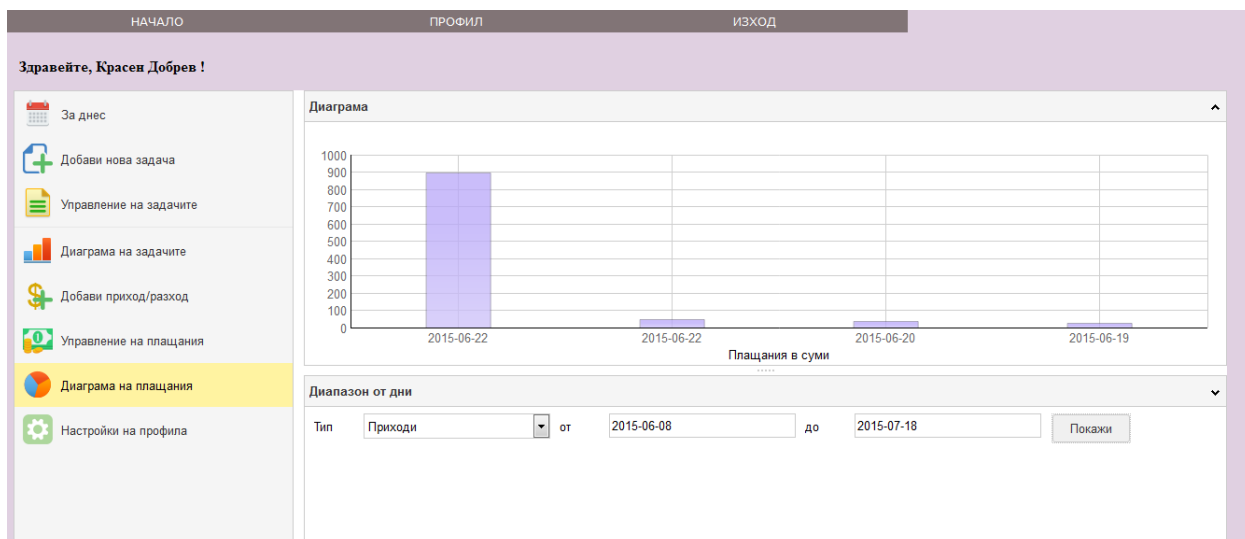
Управлявайте изгледите с помощта на филтрите.



- 1 -В това поле можете да изберете дата, за която да се покажат плащанията.
- 2- С този бутон се премахва селектираното в таблицата плащане
- 3- Бутон за обновяване на първоначалния изглед с всички плащания
- 4- Показва всички приходи
- 5- Показва всички разходи
- 6- С този филтър може да видите всички приключени плащания
- 7- При избор на този филтър се показват плащанията, които все още не са приключени.

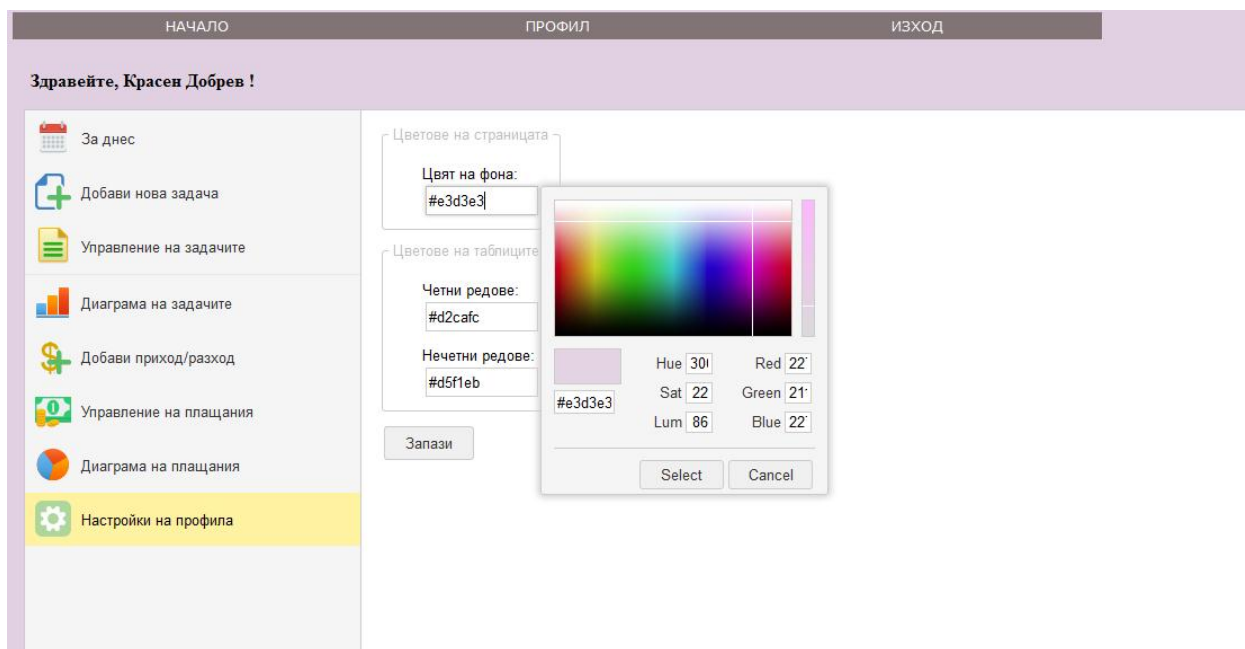
Редакцията на плащане става като се селектира ред. Данните за това плащане се визуализират във формуляра. Така можете да ги редактирате и да запазите промените с бутон „Запази“.

За да визуализирате плащанията по размери в определен период изберете раздел „Диаграма на плащания“.



Необходимо е да изберете кой вид плащания да се визуализират от поле „Тип“ и диапазона-„от“- за начална дата и „до“- за крайна дата. За да се изчертае диаграмата иберете бутона „Покажи“.

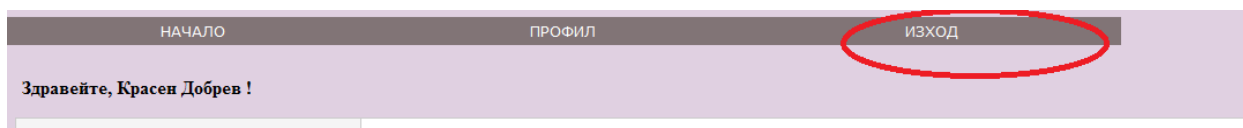
Настройка на Вашия профил



От раздел „Настройки на профила“ Вие може да промените подразбиращите се настройки за цветове. Всичко което трябва да направите е да кликнете в полето и да изберете цвят. Настройките влизат в сила след натискане на бутон „Запази“.

Изход от профила

За да излезете от профила си , изберете изход- от менюто.



След изход, ще бъдете пренасочени към страницата за вход.

5. Заключение, оценка, тестване, предложения за развитие.

Заключение

В текущото дипломно задание беше изследван процесът за разработка на уеб приложение с помощта на Java Script библиотека. Чрез създаването му бяха усвоени нови знания и практики в различни области.

В анализът на уеб приложението бяха обяснени етапите за реализиране на системата и техниките за изграждането на интерфейса. Основните функции на Java Script библиотеката dhtml бяха изпробвани, демонстрирани и документирани.

Оценка

Дипломният проект послужи за утвърждаване на познанията в доста сфери, необходими за изграждането на цялостна система.

Това което прави проекта отличителен е интерфейсът. Изграждането му беше изключително интересен процес, тъй като с разширяването на познанията си относно dhtmlx библиотеката се появяваха нови идеи за неговата организация, така, че тя да отговаря на нуждите на програмата и на операциите които се извършват с нея. Употребата на страничния раздел, на който се базира цялата функционалност, използвана от потребителя, позволи цялата логика и всички операции да се извършват на една страница, което от своя страна води до по-малък трафик и навигация.

Тестване

По време на разработката на дипломния проект бяха направени следните тестове:

- Демонстрирано беше пратането на данни от клиентската към сърварната част на приложението
- Беше демонстрирано създаването на регистрация на потребител
- Бяха съхранени данни в базата от данни
- Бяха визуализирани данни от базата с помощта на интерфейс изграден чрез dhtmlx Java Script библиотека.

- Беше демонстрирана персонализацията на интерфейса, чрез изграждане на компонентите с данни взети от базата, като цветови настройки и други
- Бяха разучени и изпробвани основните компоненти и техники за работа на dhtmlx библиотеката.

Предложения за развитие

Както всяка една система, варианти за развитие има. Тъй като в приложението се представя един от най-опростените варианти за управление на задачи, то той може да се усложни. Би могла да се реструктурира организацията като към една задача да могат да се свързват нейните подзадачи. Така при приключване на подзадача например може да се установи на колко процента е завършена главната задача. Ако една задача има четири подзадачи то при приключени две от тях тя ще е завършена на 50 %.

Тъй като всеки компонент от dhtmlx библиотеката притежава огромен набор от методи за динамично управление на визуализацията, като промяна на размер, промяна на текст, цвят, форма и шрифт на компонентите, би могло да се създаде специална таблица с настройки на интерфейса, където потребителят да може да променя всеки един параметър на dhtmlx функциите.

Приложение Б: Листинг на програмното осигуряване

JSON ФАЙЛОВЕ:

form.json

```
[
  {type: "settings", position: "label-left", labelWidth: 130, inputWidth: 300, offsetLeft: 10},
  {type: "label", label: "Нова задача"},
  {type: "input", name: "title", label: "Заглавие *", validate: "NotEmpty", offsetTop: 10},
  {type: "input", name: "description", label: "Описание", rows: 10},
  {type: "calendar", name: "date", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "Дата*",
    calendarPosition: "right"},
  {type: "checkbox", name: "urgent", label: "Неотложна"},
  {type: "button", id: "send", value: "Добави", name: "send"},
  {type: "button", id: "clear", value: "Изчисти", name: "clear"}
]
```

formPayment.json

```
[
  {type: "settings", position: "label-left", labelWidth: 130, inputWidth: 300, offsetLeft: 10},
  {type: "label", label: "Нов приход/ разход"},
  {type: "radio", name: "type", label: " Приход", labelWidth: "auto", position: "label-right",
    value: "1", checked: true},
  {type: "radio", name: "type", label: " Разход", labelWidth: "auto", position: "label-right",
```

```

value: "0"},
{type: "input", name: "description", label: "Описание", rows: 5},
{type: "calendar", name: "date", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "Дата*",
calendarPosition: "bottom"},
{type: "input", name: "amount", label: "Стойност в лева"},
{type: "checkbox", name: "urgent", label: "Неотложен"},
{type: "button", id: "send", value: "Запази", name: "send"}
]

```

formSettingsB.json

```

[
  {type: "settings", position: "label-left", labelWidth: 100, inputWidth: 100, offsetLeft: 10},
  {type: "fieldset", label: "Цветовете на страницата", list:[
    {type: "colorpicker", position:"absolute",position: "label-left",name: "background", label: "Цвят
на фона:", value: "#e3d3e3"},]},
  {type: "fieldset", position: "label-left", label: "Цветовете на таблиците", list:[
    {type: "colorpicker", position: "label-left",name: "even", label: "Четни редове:", value:
"#d2cafc"},
    {type: "colorpicker",position: "label-left", name: "uneven", label: "Нечетни редове:", value:
"#d5f1eb"}]},
  {type: "button", id: "send", value: "Запази", name: "send"}
]

```

formStatPayments.json

```
[
  {type: "settings", position: "label-left", labelWidth: 50, inputWidth: 200, offsetLeft: 10},
  {type: "select", name: "type", label: "Тип", options:[
    {text: "Приходи", value: "1"},
    {text: "Разходи", value: "0"}]},
  {type: "newcolumn"},
  {type: "calendar", name: "dateFrom", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "от",
    calendarPosition: "right"},
  {type: "newcolumn"},
  {type: "calendar", name: "dateTo", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "до",
    calendarPosition: "right"},
  {type: "newcolumn"},
  {type: "button", id: "go", value: "Покажи", name: "go"}
]
```

formStatTasks.json

```
[
  {type: "settings", position: "label-left", labelWidth: 50, inputWidth: 300, offsetLeft: 10},
  {type: "newcolumn"},
  {type: "calendar", name: "dateFrom", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "от",
```

```
    calendarPosition: "right"},
  {type: "newcolumn"},
  {type: "calendar", name: "dateTo", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "до",
    calendarPosition: "right"},
  {type: "newcolumn"},
  {type: "button", id: "go", value: "Покажи", name: "go"}
]
```

formView.json

```
[
  {type: "settings", position: "label-left", labelWidth: 130, inputWidth: 300, offsetLeft: 10},
  {type: "label", label: "Нова задача"},
  {type: "input", name: "title", label: "Заглавие *", validate: "NotEmpty", offsetTop: 10},
  {type: "input", name: "description", label: "Описание", rows: 10},
  {type: "newcolumn"},
  {type: "input", name: "id", offsetTop: 10},
  {type: "calendar", name: "date", dateFormat: "%Y-%m-%d", label: "Дата", calendarPosition: "bottom"},
  {type: "checkbox", name: "urgent", label: "Неотложна"},
  {type: "checkbox", name: "completed", label: "Завършена"},
  {type: "button", id: "send", value: "Запази", name: "save"}
]
```


formViewPayments.json

```
[
    {type: "settings", position: "label-left", labelWidth: 130, inputWidth: 300, offsetLeft: 10},
    {type: "label", label: "Преглед на плащане"},
    {type: "radio", name: "type", label: " Приход", labelWidth: "auto", value: "1"},
    {type: "radio", name: "type", label: " Разход", labelWidth: "auto", value: "0"},
    {type: "input", name: "description", label: "Описание *", validate: "NotEmpty", rows: 10},
    {type: "newcolumn"},
    {type: "input", name: "id", offsetTop: 10},
    {type: "calendar", name: "date", dateFormat: "%Y-%m-%d", validate: "NotEmpty", label: "Дата*",
    calendarPosition: "bottom"},
    {type: "input", name: "amount", validate: "NotEmpty", label: "Размер*"},
    {type: "checkbox", name: "completed", label: "Приключен"},
    {type: "checkbox", name: "urgent", label: "Неотложен"},
    {type: "button", id: "send", value: "Запази", name: "save"}
]
```

registration.json

```
[
    {type: "settings", position: "label-left", labelWidth: 300, inputWidth: 300},
    {type: "fieldset", label: "Добре дошли!", inputWidth: 680, list:[
    {type: "input", name: "tFirst", label: "Име *", validate: "NotEmpty"},
    {type: "input", name: "tLast", label: "Фамилия *", validate: "NotEmpty"},
    ]
}
```

```

    {type: "input",      name: "tMail",  label: "Ел. поща *",    validate: "NotEmpty,ValidEmail", note:
    {text: "Моля, въведете валиден адрес!"}},
    {type: "password",  name: "pPass",  label: "Парола *",      validate: "NotEmpty"},
    {type: "password",  name: "pPass2", label: "Повтори парола *", validate: "NotEmpty"},
    {type: "button",    name: "send" ,  id: "send",              value: "Добави" }
  ]]
]

```

sidebar.json

```

{items: [
  {id: "today",          text: "За днес",          icon: "today.png", selected: true},
  {id: "addTask",        text: "Добави нова задача",  icon: "add.png"},
  {id: "manageTasks",    text: "Управление на задачите", icon: "documents.png"},
  {type: "separator"},
  {id: "day",            text: "Диаграма на задачите",  icon: "chart.png"},
  {id: "addInOut",        text: "Добави приход/разход",  icon: "addMoney2.png"},
  {id: "managePayments", text: "Управление на плащания", icon: "money.png"},
  {id: "calculate",      text: "Диаграма на плащания",  icon: "pie.png"},
  {id: "settings",       text: "Настройки на профила",  icon: "settings.png"},
]]

```

XML ФАЙЛОВЕ:

chartPayments.xml

```
<?xml version="1.0"?>
<data>
    <item id="1">
        <amount>900</amount>
        <date>2015-06-22</date>
    </item>
    <item id="2">
        <amount>50</amount>
        <date>2015-06-22</date>
    </item>
    <item id="3">
        <amount>40</amount>
        <date>2015-06-20</date>
    </item>
    <item id="4">
        <amount>30</amount>
        <date>2015-06-19</date>
    </item>
</data>
```

chartTasks.xml

```
<?xml version="1.0"?>
```

```
<data>
    <item id="1">
        <taskscount>1</taskscount>
        <date>2015-06-19</date>
    </item>
    <item id="2">
        <taskscount>2</taskscount>
        <date>2015-06-28</date>
    </item>
    <item id="3">
        <taskscount>3</taskscount>
        <date>2015-06-22</date>
    </item>
    <item id="4">
        <taskscount>2</taskscount>
        <date>2015-06-21</date>
    </item>
</data>
```

toolbarPayments.xml

```
<?xml version="1.0"?>
<toolbar>
```

```

<item id="delete" type="button" img="close.gif" imgdis="open_dis.gif" text="Премахни" />
<item id="reload" type="button" img="reload.gif" imgdis="save_dis.gif" text="Обнови" />
<item id="sep2" type="separator"/>
<item id="incomes" type="button" img="plus1.png" imgdis="cut_dis.gif" text="Приходи" />
<item id="outcomes" type="button" img="minus.png" imgdis="copy_dis.gif" text="Разходи" />
<item id="filterFinished" type="button" img="ok.gif" imgdis="copy_dis.gif" text="Приключени"
/>
<item id="filterUnfinished" type="button" img="list.png" imgdis="copy_dis.gif"
text="Неприключени" />
</toolbar>

```

toolbarTaskStat.xml

<?xml version="1.0"?>

```

<toolbar>
  <item id="delete" type="button" img="close.gif" imgdis="open_dis.gif" text="Премахване" />
  <item id="reload" type="button" img="reload.gif" imgdis="save_dis.gif" text="Обнови" />
  <item id="sep1" type="separator"/>
  <item id="filterUrgent" type="button" img="urgent.png" imgdis="cut_dis.gif" text="Задължителни"
/>
  <item id="filterOptional" type="button" img="optional.png" imgdis="copy_dis.gif"
text="Незадължителни" />
  <item id="filterFinished" type="button" img="ok.gif" imgdis="copy_dis.gif" text="Приключени" />
  <item id="filterUnfinished" type="button" img="list.png" imgdis="copy_dis.gif"

```

```
        text="Неприключени"  />
</toolbar>
```

toolbarToday.xml

```
<?xml version="1.0"?>
<toolbar>
    <item id="complete"    type="button"    img="ok.gif"    imgdis="save_dis.gif"    text="Отбележи като
приключена"  />
    <item id="delete"      type="button"    img="close.gif"    imgdis="open_dis.gif"    text="Премахване"  />
</toolbar>
```

toolbarTodayPayments.xml

```
<?xml version="1.0"?>
<toolbar>
    <item id="complete"    type="button"    img="ok.gif"    imgdis="save_dis.gif"    text="Отбележи като
приключена"  />
    <item id="delete"      type="button"    img="close.gif"    imgdis="open_dis.gif"    text="Премахване"  />
</toolbar>
```

toolbarUserPage.xml

```
<?xml version="1.0"?>
<toolbar>
```

```
<item id="delete" type="button" img="close.gif" imgdis="open_dis.gif" text="Премахване" />
<item id="reload" type="button" img="reload.gif" imgdis="save_dis.gif" text="Обнови" />
<item id="sep1" type="separator"/>
<item id="filterUrgent" type="button" img="urgent.png" imgdis="cut_dis.gif"
text="Задължителни" />
<item id="filterOptional" type="button" img="optional.png" imgdis="copy_dis.gif"
text="Незадължителни" />
<item id="filterFinished" type="button" img="ok.gif" imgdis="copy_dis.gif" text="Приключени"
/>
<item id="filterUnfinished" type="button" img="list.png" imgdis="copy_dis.gif"
text="Неприключени" />
</toolbar>
```

RНР ФАЙЛОВЕ:

deletePayment.php

```
<?php

session_start();
include('setup.php');
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    include('setup.php');
```

```
$id = $_POST['paymentId'];

$qDELETE = "DELETE FROM payments WHERE id='$id'";
if (mysqli_query($dbc, $qDELETE)) {

    if (!mysqli_commit($dbc)) {
        print("Transaction commit failed\n");
        exit();
    }

    echo "Плащането е изтрито.";
} else {
    echo "Error: " . $qDELETE . "<br>" . mysqli_error($dbc);
}
mysqli_close($dbc);
include('updateXMLs.php');

}

?>
```

deleteTask.php


```
<?php

session_start();
include('setup.php');
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    include('setup.php');
    $id = $_POST['taskId'];

    $qDELETE = "DELETE FROM usertasks WHERE id='$id'";

    if (mysqli_query($dbc, $qDELETE)) {

        if (!mysqli_commit($dbc)) {
            print("Transaction commit failed\n");
            exit();
        }

        echo "Задачата е изтрита.";
    } else {
        echo "Error: " . $qDELETE . "<br>" . mysqli_error($dbc);
    }
}
```

```
mysqli_close($dbc);  
include('updateXMLs.php');  
  
}  
?>
```

getPayments.php

```
<?php
```

```
$userid=$_SESSION['userid'];  
include('setup.php');
```

```
$q = "SELECT * FROM payments WHERE userid='$userid'";  
$r = mysqli_query($dbc, $q);  
//$page = mysqli_fetch_assoc($r);
```

```
$xml("<rows> \n \t");  
$xml.="<head> \n \t\t" ;  
$xml.="<column width=\"70\" type=\"ro\" align=\"left\" sort=\"str\"> Homep </column> \n \t\t";  
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Вид </column> \n \t\t";  
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Дата </column> \n \t\t";
```

```

$xml.="<column width=\"300\" type=\"ro\" align=\"left\" sort=\"str\"> Описание </column> \n \t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Размер </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Задължителна </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Приключена </column> \n \t";
$xml.="</head>\n";
//$xml="";

$num=1;
while($page = mysqli_fetch_assoc($r))
{
    if($page['urgent']==1) $urgent="Да"; else $urgent="Не";
    if($page['completed']==1) $completed="Да"; else $completed="Не";
    if($page['type']==1) $type="Приход"; else $type="Разход";

    $xml .=" \t <row id=\"".$num.">\n \t\t";
    $xml .="<cell>".$page['id']."</cell>\n\t\t";
    $xml .="<cell>".$type."</cell>\n\t\t";
    $xml .="<cell>".$page['date']."</cell>\n\t\t";
    $xml .="<cell>".$page['description']."</cell>\n\t\t";
    $xml .="<cell>".$page['amount']."</cell>\n\t\t";
    $xml .="<cell>".$urgent."</cell>\n\t\t";
    $xml .="<cell>".$completed."</cell>\n\t\t";
    $xml.="</row>\n\t\t";
}

```

```

        $num++;
    }

    $xml.="</rows>";
    $xmlobj=new SimpleXMLElement($xml);
    $xmlobj->asXML("xml/gridPayments.xml");
    mysqli_close($dbc);

?>

```

getSettings.php

```

<?php
session_start();
include('setup.php');
include('globals.php');
$userid=$_SESSION['userid'];

    $qSettings = "SELECT * FROM userssettings where userid='$userid'";
    $rSettings = mysqli_query($dbc, $qSettings);
    $userdataSettings= mysqli_fetch_assoc($rSettings);

    $_SESSION['background'] = $userdataSettings['background'];
    $_SESSION['even']=$userdataSettings['even'];

```

```
$_SESSION['uneven']=$userdataSettings['uneven'];  
?>
```

getStatPayments.php

```
<?php  
session_start();  
  
$userid=$_SESSION['userid'];  
include('setup.php');  
echo $_SESSION['userid'];  
  
$dateFrom = $_POST['dateFrom'];  
$dateTo = $_POST['dateTo'];  
$type = $_POST['type'];  
$q="SELECT *  
    FROM payments  
    WHERE (date BETWEEN '$dateFrom' AND '$dateTo') AND  userid='$userid' AND type='$type'";  
  
$r = mysqli_query($dbc, $q);  
$num=1;  
$xml="<data> \n \t";  
  
print_r($_POST);
```

```

while ($page = mysqli_fetch_assoc($r)) {

    $xml .= " \t <item id=\"\".$num.\">\n \t\t";
    $xml .= "<amount>".$page['amount']."</amount>\n\t\t";
    $xml .= "<date>". $page['date']."</date>\n\t";
    $xml.="</item>\n\t";
    $num++;
}

$xml.="</data>";
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/chartPayments.xml");
mysqli_close($dbc);

```

getStatTasks.php

```

<?php
session_start();

$userid=$_SESSION['userid'];
include('setup.php');
echo $_SESSION['userid'];

```

```

$dateFrom = $_POST['dateFrom'];
$dateTo = $_POST['dateTo'];
$q="SELECT *
    FROM usertasks
    WHERE (date BETWEEN '$dateFrom' AND '$dateTo') AND  userid='$userid'";

$r = mysqli_query($dbc, $q);
$num = mysqli_num_rows($r);

print_r($num);
$arrayAllDates = array();
while ($page = mysqli_fetch_assoc($r)) {
    array_push($arrayAllDates, $page['date']);
}

print_r($arrayAllDates);
$counts=array();
$counts=array_count_values($arrayAllDates);
    print_r($counts);
$num=1;
$xml("<data> \n \t";

foreach ($counts as $key => $value) {

```

```

$xml .= " \t <item id=\"\".$num.\">\n \t\t";
$xml .= "<taskcount>".$value."</taskcount>\n\t\t";
$xml .= "<date>". $key."</date>\n\t";
$xml.="</item>\n\t";
$num++;
}
$xml.="</data>";
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/chartTasks.xml");
mysqli_close($dbc);

```

getTasks.php

```

<?php

$userid=$_SESSION['userid'];
include('setup.php');

$q = "SELECT * FROM usertasks WHERE userid='$userid'";
$r = mysqli_query($dbc, $q);

$xml="<rows> \n \t";

```



```

$xml.="<head> \n \t\t" ;
$xml.="<column width=\"70\" type=\"ro\" align=\"left\" sort=\"str\"> Номер </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Дата </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Заглавие </column> \n \t";
$xml.="<column width=\"400\" type=\"ro\" align=\"left\" sort=\"str\"> Описание </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Задължителна </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Приключена </column> \n \t";
$xml.="</head>\n";
//$xml="";

$num=1;
while($page = mysqli_fetch_assoc($r))
{

    if($page['urgent']==1) $urgent="Да"; else $urgent="Не";
    if($page['completed']==1) $completed="Да"; else $completed="Не";
    if($page['date']=="0000-00-00") $date="Безсрочна"; else $date=$page['date'];
    $xml .=" \t <row id=\"".$num."\">\n \t\t";
    $xml .=" <cell>".$page['id']. "</cell>\n\t\t";
    $xml .=" <cell>".$date."</cell>\n\t\t";
    $xml .=" <cell>".$page['title']. "</cell>\n\t\t";
    $xml .=" <cell>".$page['description']. "</cell>\n\t\t";
    $xml .=" <cell>".$urgent."</cell>\n\t\t";
    $xml .=" <cell>".$completed."</cell>\n\t\t";
}

```

```
$xml.="</row>\n\t";
$num++;
}
$xml.="</rows>";
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/text.xml");

mysqli_close($dbc);

?>
```

getTodayPayments.php

```
<?php

$userid=$_SESSION['userid'];
include('setup.php');

$q = "SELECT * FROM payments WHERE DATE= CURDATE() AND userid='$userid'";
$r = mysqli_query($dbc, $q);

$xml="<rows> \n \t";
```

```

$xml.="<head> \n \t\t" ;
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Номер </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Вид </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Дата </column> \n \t\t";
$xml.="<column width=\"400\" type=\"ro\" align=\"left\" sort=\"str\"> Описание </column> \n \t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Размер </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Задължителен </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Приключен </column> \n \t";
$xml.="</head>\n";
//$xml="";

$num=1;
while($page = mysqli_fetch_assoc($r))
{

    if($page['urgent']==1) $urgent="Да"; else $urgent="Не";
    if($page['completed']==1) $completed="Да"; else $completed="Не";
    if($page['type']==1) $type="Приход"; else $type="Разход";

    $xml .=" \t <row id=\"".$num.">\n \t\t";
    $xml .=" <cell>".$page['id']."</cell>\n\t\t";
    $xml .=" <cell>".$type."</cell>\n\t\t";
    $xml .=" <cell>".$page['date']."</cell>\n\t";
}

```

```

$xml .= "<cell>".$page['description']. "</cell>\n\t";
$xml .= "<cell>".$page['amount']. "</cell>\n\t\t";
$xml .= "<cell>".$urgent. "</cell>\n\t";
$xml .= "<cell>".$completed. "</cell>\n\t";
$xml.="</row>\n\t";
$num++;

}
$xml.="</rows>";
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/todayPayments.xml");
mysqli_close($dbc);

```

?>

getTodayTasks.php

```
<?php
```

```

$userid=$_SESSION['userid'];
include('setup.php');

```

```
$q = "SELECT * FROM usertasks WHERE DATE=CURDATE() AND userid='$userid'";
```

```

$r = mysqli_query($dbc, $q);
//$page = mysqli_fetch_assoc($r);

$xml=<"<rows> \n \t";
$xml.="<head> \n \t\t" ;
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"int\"> Номер </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Дата </column> \n \t\t";
$xml.="<column width=\"100\" type=\"ro\" align=\"left\" sort=\"str\"> Заглавие </column> \n \t";
$xml.="<column width=\"400\" type=\"ro\" align=\"left\" sort=\"str\"> Описание </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Задължителна </column> \n \t";
$xml.="<column width=\"150\" type=\"ro\" align=\"left\" sort=\"str\"> Приключена </column> \n \t";
$xml.="</head>\n";
//$xml="";

$num=1;
while($page = mysqli_fetch_assoc($r))
{

    if($page['urgent']==1) $urgent="Да"; else $urgent="Не";
    if($page['completed']==1) $completed="Да"; else $completed="Не";
    if($page['date']=="0000-00-00") $date="Безсрочна"; else $date=$page['date'];
    $xml .=" \t <row id=\"".$num.">\n \t\t";
    $xml .=" <cell>".$page['id']."</cell>\n\t\t";

```

```

$xml .= "<cell>".$date."</cell>\n\t\t";
$xml .= "<cell>".$page['title']."</cell>\n\t\t";
$xml .= "<cell>".$page['description']."</cell>\n\t";
$xml .= "<cell>".$urgent."</cell>\n\t";
$xml .= "<cell>".$completed."</cell>\n\t";
$xml.="</row>\n\t";
$num++;

}

$xml.="</rows>";
$xmlobj=new SimpleXMLElement($xml);
$xmlobj->asXML("xml/todayTasks.xml");
mysqli_close($dbc);

?>

```

index.php

```
<?php
```

```
#Старт на сесия
```

```
session_start();
include('menu.php');

?>
<html>
<head>
  <style>
    .basic {
      position: absolute;
      left: 300px;
      top:100px;
      width: 700px;
      font-family: Verdana, Geneva, sans-serif;
      font-size: 24px;
      color: white;
      text-align: center;
    }

    .right {
      position: absolute;
      right: 0px;
      top:530px;
      width: 680px;
      background-color: #75d03b;
```

```

        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;
    }
</style>
</head>
<body background="common/imgs/financial-planning1366dark.jpg">

    <div class="basic">
        <p>
            &nbsp; &nbsp; &nbsp; Органайзърът е инструмент за управление на задачи и плащания, който Ви предлага:
        <ul>
            <li>Съхранение на задачи и плащания</li>
            <li>Управление на задачи и плащания</li>
            <li>Диаграми и изгледи</li>
            <li>Персонализация на Вашия профил</li>
        </ul>
        <br>
        <?php if((!isset($_SESSION['username'])))
        {
            echo " &nbsp; &nbsp; &nbsp; За да използвате приложението е необходимо да създадете Ваша <a
href=\"registration.php\" > регистрация </a>.";
        }

        ?>
    </p>

```



```
        </div>
    <div class="right">
        <p> &nbsp; &nbsp; <a href="userPage.php">Напред към приложението!</a></p>
    </div>
</ul>

</body>
</html>
```

login.php

```
<?php
session_start();
include('setup.php');

#Проверка за съответствие между потребител и парола

if ($_POST) {

    $q = "SELECT * FROM users where email='$_POST[user]' AND password='$_POST[password]'";
    $r = mysqli_query($dbc, $q);
    $num = mysqli_num_rows($r);
    $userdata = mysqli_fetch_assoc($r);
```

```

if ($num == 1) {
    $_SESSION['username'] = $_POST['user'];
    $_SESSION['userid'] = $userdata['id'];
    $_SESSION['userfirst'] = $userdata['first'];
    $_SESSION['userlast'] = $userdata['last'];
    $_SESSION['background'] = $userdata['background'];
    $_SESSION['even'] = $userdata['even'];
    $_SESSION['uneven'] = $userdata['uneven'];
    header('Location: userPage.php');
} else {
    echo "<script> alert (\"Невалидни данни!\") </script>";
}
}
mysqli_close($dbc);
?>

<html>
<head>
    <title>Вход</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <script src="codebase/dhtmlx.js"></script>

```

```
<link rel="stylesheet" type="text/css" href="css/login.css"/>
<style>
    .rightt {
        position: absolute;
        left: 0px;
        top: 30px;
        width: 400px;
        background-color: #9bb7df;
        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;
    }

    .left {
        position: absolute;
        right: 0px;
        top: 500px;
        width: 400px;
        background-color: #9bb7df;
        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;
    }

</style>
```

```
</head>

<body onload="doOnLoad();" >
  <?php include('menu.php'); ?>
  <h2>Влезте в профила си, за да използвате приложението!</h2>
  <div class="signup">
    <form action="login.php" method="post" role="form" >
      <span class="ribben">Вход в системата</span>
      <p>Електронен адрес : <span class="dot"> </span></p>
      <input type="text" name="user" placeholder="example@email.com" required="">
      <p>Парола: <span class="dot"> </span></p>
      <input type="password" name="password" placeholder="" required="">

      <input type="submit" value="Вход">
    </form>
  </div>

</body>
</html>
```

logout.php

```
<?php

#Старт на сесия

if(!isset($_SESSION['username'])) {
    header('Location: login.php');
}

session_start();
#Изтриване на username
unset($_SESSION['username']);
#Изтрива всички ключове
session_destroy();
header('Location: login.php');
?>
```

menu.php

```
<style>
    ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
```

```
        overflow: hidden;

    }

    li {
        float: left;
        display: inline;
        width: 320px;
        height: 50px;
        font-size: 22px;
    }

    ul a:link, ul a:visited {
        display: block;
        width: 320px;
        font-family: Verdana, Geneva, sans-serif, bold;
        font-size: 60%;
        color: #FFFFFF;
        background-color: #817476;
        text-align: center;
        padding: 4px;
        text-decoration: none;
        text-transform: uppercase;
    }
}
```

```

}

ul a:hover, ul a:active {
    background-color: #9bb7df;
}
</style>
<ul>
    <li id="liMenu"><a href="index.php">Начало                </a></li>
    <li id="liMenu" ><a href="userPage.php" id>Профил          </a></li>
    <?php if(isset($_SESSION['username'])) echo  "<li id=\"liMenu\"><a href=\"logout.php\">Исход
</a></li>";
    else{
        echo"
        <li id=\"liMenu\" color=\"red\"><a href=\"login.php\" id>Вход                </a></li>
        <li id=\"liMenu\" color=\"red\"><a href=\"registration.php\" id>Регистрация
</a></li>";
    }
?>

</ul>

registerme.php

<?php

```

```

include('setup.php');
$emails = array();
$q = "SELECT email FROM users ";
$r = mysqli_query($dbc, $q);

while ($row = mysqli_fetch_assoc($r)) {
    $emails[] = $row['email'];
}
if (!in_array($_POST['tMail'], $emails)) {
    $first = $_POST['tFirst'];
    $last = $_POST['tLast'];
    $email = $_POST['tMail'];
    $psw = $_POST['pPass'];

    $qINSERT = "INSERT INTO users (first,last,email,password,background,even, uneven)
                VALUES
('$first','$last','$email','$psw','#e3d3e3','#d2cafc','#d5f1eb');"

    if (mysqli_query($dbc, $qINSERT)) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $qINSERT . "<br>" . mysqli_error($dbc);
    }
}

```



```
} else {  
    echo "Потребител с този адрес вече съществува!";  
}
```

registration.php

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title> Регистрация </title>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>  
        <meta http-equiv="X-UA-Compatible" content="IE=edge"/>  
        <link rel="stylesheet" type="text/css" href="codebase/dhtmlx.css"/>  
        <script src="codebase/dhtmlx.js"></script>  
        <style>  
            .left {  
                position: absolute;  
                left: 0px;  
                top: 60px;  
                width: 400px;  
                background-color: #87e14e;
```

```
        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;
    }

    .right {
        position: absolute;
        right: 0px;
        top: 530px;
        width: 400px;
        background-color: #87e14e;
        font-family: Verdana, Geneva, sans-serif;
        font-size: 12px;
    }

</style>
<script>
    var myForm, formData;
    function doOnLoad() {
        myForm = new dhtmlXForm("myForm");
        myForm.loadStruct("json/registration.json", function () {
        });
        myForm.setFontSize("15px");
    }
</script>
```

```

myForm.attachEvent("onButtonClick", function (id) {
    if (id == "send") {
        alert("button clicked");
        if (myForm.validate()) {
            if (myForm.getItemValue("pPass") == myForm.getItemValue("pPass2")) {
                myForm.send("registerme.php", "post", function (loader, response) {
                    alert(response);
                });
            }
            else {
                alert("Паролите не съвпадат!");
                myForm.setItemValue("pPass", "");
                myForm.setItemValue("pPass2", "");
            }
        }
    }
});

}

function clearmyForm()
{
    myForm.setItemValue("tFirst", "");
    myForm.setItemValue("tLast", "");
}

```

```
        myForm.setItemValue("tMail", "");  
        myForm.setItemValue("pPass", "");  
        myForm.setItemValue("pPass2", "");  
    }  
  
</script>  
  
</head>  
<body onload="doOnLoad();" background="common/imgs/signing.jpg">  
    <?php include('menu.php') ?>  
    <div id="myForm" style="height:100px; position:relative; left: 340px; top: 140px" ></div>  
    <div class="left">  
        <p>    Регистрирайте се за да използвате приложението.</p>  
  
    </div>  
  
    <div class="right">  
        <p>    ;    ;<a href="login.php">Напред към страницата за вход!</a></p>  
  
    </div>  
</body>  
</html>
```

savePayment.php

```
<?php
```

```
session_start();
```

```
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {  
    include('setup.php');
```

```
    $type=$_POST['type'];  
    $desc = $_POST['description'];  
    $date=$_POST['date'];  
    $urgent=$_POST['urgent'];  
    $userid=$_SESSION['userid'];  
    $amount=$_POST['amount'];  
    $type=$_POST['type'];
```

```
$qINSERT = "INSERT INTO payments (description,date,urgent,userid,amount,type)  
            VALUES ('$desc','$date','$urgent','$userid','$amount','$type')";
```

```
if (mysqli_query($dbc, $qINSERT)) {
```

```

        if (!mysqli_commit($dbc)) {
            print("Transaction commit failed\n");
            exit();
        }

        echo "Плащането е добавено!";
    } else {
        echo "Error: " . $qINSERT . "<br>" . mysqli_error($dbc);
    }
    mysqli_close($dbc);
    include('updateXMLs.php');

}
?>

```

saveTask.php

```

<?php

session_start();

if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {

    include('setup.php');

```

```

$title = $_POST['title'];
$desc = $_POST['description'];
$date=$_POST['date'];
$urgent=$_POST['urgent'];
$userid=$_SESSION['userid'];

$qINSERT = "INSERT INTO usertasks (title,description,date,urgent,userid)
            VALUES ('$title','$desc','$date','$urgent','$userid');"

if (mysqli_query($dbc, $qINSERT)) {

    if (!mysqli_commit($dbc)) {
        print("Transaction commit failed\n");
        exit();
    }

    echo "Задачата беше добавена успешно!";
} else {
    echo "Error: " . $qINSERT . "<br>" . mysqli_error($dbc);
}

mysqli_close($dbc);
include('updateXMLs.php');

```

```
}
```

```
?>
```

setPaymentCompleted.php

```
<?php
```

```
session_start();
```

```
print_r($_POST);
```

```
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {
```

```
    include('setup.php');
```

```
    $paymentid=$_POST['paymentId'];
```

```
    $qUPDATE = "UPDATE payments SET completed=true WHERE id='$paymentid'";
```

```
    if (mysqli_query($dbc, $qUPDATE)) {
```

```
        if (!mysqli_commit($dbc)) {
```

```
            print("Transaction commit failed\n");
```

```
            exit();
```



```

        }
        echo "New record updated successfully";
    } else {
        echo "Error: " . $qUPDATE . "<br>" . mysqli_error($dbc);
    }
    mysqli_close($dbc);
    include('updateXMLs.php');
}

```

?>

setTaskCompleted.php

```
<?php
```

```
session_start();
```

```
print_r($_POST);
```

```
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {
```

```
    include('setup.php');
```

```

        $taskid=$_POST['taskId'];
        $qUPDATE = "UPDATE usertasks SET completed=true WHERE id='$taskid'";

        if (mysqli_query($dbc, $qUPDATE)) {

            if (!mysqli_commit($dbc)) {
                print("Transaction commit failed\n");
                exit();
            }

            echo "New record updated successfully";
        } else {
            echo "Error: " . $qUPDATE . "<br>" . mysqli_error($dbc);
        }
        mysqli_close($dbc);
        include('updateXMLs.php');
    }

?>

```

setUser.php

```

<?php
session_start();

```

```

if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {

    include('setup.php');
    $pass=$_POST['password'];
    $usr=$_POST['user'];
    $q = "SELECT * FROM users where email='$_POST[user]' AND password='$_POST[password]'";
    $r = mysqli_query($dbc, $q);
    $num = mysqli_num_rows($r);
    if ($num == 1) {
        echo "Yes";
        $_SESSION['username'] =$_POST[user];
        $_SESSION['username'] = $_POST['user'];
        $_SESSION['userid']=$userdata['id'];
        $_SESSION['userfirst']=$userdata['first'];
        $_SESSION['userlast']=$userdata['last'];

        header('Location: index.php');

    } else {
        echo "Моля, проверете отново името и паролата си!";
    }
}

?>

```

setup.php

```
<?php
$dbc = mysqli_connect('localhost', 'dev', 'loapsw', 'budget_planning') OR die('Error' .
mysqli_connect_errno());
?>
```

updatePayments.php

```
<?php

print_r($_POST);
session_start();
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {

    include('setup.php');

    $desc = $_POST['description'];
    $type = $_POST['type'];
    $date=$_POST['date'];
    $urgent=$_POST['urgent'];
    $completed=$_POST['completed'];
```

```
$paymentid=$_POST['id'];  
$amount=$_POST['amount'];
```

```
$qUPDATE = "UPDATE payments SET  
description='$desc',type='$type',date='$date',amount='$amount',urgent='$urgent',completed='$completed' WHERE  
id='$paymentid'";
```

```
if (mysqli_query($dbc, $qUPDATE)) {  
  
    if (!mysqli_commit($dbc)) {  
        print("Transaction commit failed\n");  
        exit();  
    }  
    echo "New record updated successfully";  
} else {  
    echo "Error: " . $qUPDATE . "<br>" . mysqli_error($dbc);  
}  
mysqli_close($dbc);
```

```
include('updateXMLs.php');  
}
```

?>

updateSettings.php

<?php

```
print_r($_POST);
```

```
session_start();
```

```
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {
```

```
    include('setup.php');
```

```
        $userid=$_SESSION['userid'];
```

```
        $background = $_POST['background'];
```

```
        $even = $_POST['even'];
```

```
        $uneven=$_POST['uneven'];
```

```
        $qUPDATE = "UPDATE users SET
```

```
background='$background',even='$even',uneven='$uneven' WHERE id='$userid';";
```

```
        if (mysqli_query($dbc, $qUPDATE)) {
```

```

        if (!mysqli_commit($dbc)) {
            print("Transaction commit failed\n");
            exit();
        }
        echo "New record updated successfully";
    } else {
        echo "Error: " . $qUPDATE . "<br>" . mysqli_error($dbc);
    }
    mysqli_close($dbc);
    $_SESSION['background']=$background;
    $_SESSION['even']=$even;
    $_SESSION['uneven']=$uneven;
}

```

updateTask.php

```

<?php

session_start();
print_r($_POST);

if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {

```

```
include('setup.php');
```

```
$title = $_POST['title'];  
$desc = $_POST['description'];  
$date=$_POST['date'];  
$urgent=$_POST['urgent'];  
$completed=$_POST['completed'];  
$userid=$_SESSION['userid'];  
$taskid=$_POST['id'];
```

```
$qUPDATE = "UPDATE usertasks SET title='$title',description='$desc',  
date='$date',urgent='$urgent',completed='$completed' WHERE id='$taskid'";
```

```
if (mysqli_query($dbc, $qUPDATE)) {
```

```
    if (!mysqli_commit($dbc)) {  
        print("Transaction commit failed\n");  
        exit();  
    }  
    echo "New record updated successfully";
```

```
} else {  
    echo "Error: " . $qUPDATE . "<br>" . mysqli_error($dbc);
```



```
    }  
    mysqli_close($dbc);  
    include('updateXMLs.php');  
}
```

?>

updateXMLs.php

<?php

```
include('getTasks.php');  
include('getTodayTasks.php');  
include('getPayments.php');  
include('getTodayPayments.php');
```

userPage.php

<?php

```
session_start();
```

```
if (!isset($_SESSION['username'])) {
```

```
header('Location: login.php');
}
?>
<html>
  <head>

    <title>Profile</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <link rel="stylesheet" type="text/css" href="skins/terrace/dhtmlx.css"/>

    <script src="codebase/dhtmlx.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <script src="functions.js"></script>
    <meta charset="UTF-8">

  <style>
    div#sidebarObj {
      position: relative;
      margin-left: 10px;
      margin-top: 10px;
      width: 1300px;
      height: 600px;
```

```

    }
</style>

<style>
    .even{
        background-color:<?php echo $_SESSION['even'];?> ; //. " " . $_SESSION['userlast']
    }
    .uneven{
        background-color:<?php echo $_SESSION['uneven']; ?>; // #F0F8FF;
    }
</style>

<style>
    div#layoutObjChartTasks {
        position: relative;
        margin-top: 20px;
        margin-left: 20px;
        width: 600px;
        height: 400px;
    }
</style>

<script>
    var mySidebar;

```

```
var myForm;
```

```
function doOnLoad() {
```

```
    mySidebar = new dhtmlXSideBar({  
        parent: "sidebarObj",  
        icons_path: "common/win_32x32/",  
        template: "tiles",  
        width: 300,  
        json: "json/sidebar.json",  
        onload: function () {
```

```
            //ЗА ДНЕС
```

```
            myLayoutToday = mySidebar.cells("today").attachLayout({  
                pattern: "2E",  
                cells: [{id: "a", text: "Задачи"}, {id: "b", text: "Планирование"}]  
            });  
            myMenuTodayTasks = myLayoutToday.cells("a").attachToolbar({  
                icons_path: "common/imgs/",  
                xml: "xml/toolbarToday.xml"
```

```

});

myMenuTodayPayments = myLayoutToday.cells("b").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarToday.xml"
});

myGridTodayTasks = myLayoutToday.cells("a").attachGrid();
myGridTodayTasks.setImagePath("imgs/");
<?php include('getTodayTasks.php'); ?>
myGridTodayTasks.loadXML("xml/todayTasks.xml");
myGridTodayTasks.enableAlterCss("even", "uneven");

myMenuTodayTasks.attachEvent("onClick", function (id) {
    if (id == "delete")
    {
        var selectionTodayTasks=myGridTodayTasks.getSelectedRowId();
        var valueCellTodayTasks=
            myGridTodayTasks.cells(selectionTodayTasks,0).getValue();
        deleteTask(valueCellTodayTasks);
        updateGrids();
    }
    if (id == "complete")
    {

```

```

        var selectionTodayTasks=myGridTodayTasks.getSelectedRowId();
        var valueCellTodayTasks=
        myGridTodayTasks.cells(selectionTodayTasks,0).getValue();
        completeTask(valueCellTodayTasks);
        updateGrids();
    }

});

```

```

myGridTodayPayments = myLayoutToday.cells("b").attachGrid();
myGridTodayPayments.setImagePath("imgs/");
<?php include('getTodayPayments.php'); ?>
myGridTodayPayments.loadXML("xml/todayPayments.xml");
myGridTodayPayments.enableAlterCss("even", "uneven");
myGridTodayPayments.attachEvent("onRowSelect", doOnRowSelectedTodayPayments);
myMenuTodayPayments.attachEvent("onClick", function (id) {
    if (id == "delete")
    {
        var selectionTodayPayments = myGridTodayPayments.getSelectedRowId();
        var valueCellTodayPayments =
        myGridTodayPayments.cells(selectionTodayPayments, 0).getValue();
    }
});

```

```

        deletePayment(valueCellTodayPayments);
        updateGrids();
    }
    if (id == "complete")
    {
        var selectionTodayPayments = myGridTodayPayments.getSelectedRowId();
        var valueCellTodayPayments =
        myGridTodayPayments.cells(selectionTodayPayments, 0).getValue();
        completePayment(valueCellTodayPayments);
        updateGrids();
    }

});

//Нова задача

myForm = mySidebar.cells("addTask").attachForm();
myForm.loadStruct("json/form.json");
myForm.attachEvent("onButtonClick", function (id) {
    if (id == "send") {
        myForm.send("saveTask.php", "post", function (loader, response) {
            alert(response);
            updateGrids();
        });
    }
});

```

```

        clearmyForm();

    });

}

if (id == "clear") {
    clearmyForm();
}

});

//УПРАВЛЕНИЕ НА ЗАДАЧИ

myLayoutManage = mySidebar.cells("manageTasks").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Задачи", height: "190"}, {id: "b", text: "Преглед"}]
});

myMenu = mySidebar.cells("manageTasks").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarUserPage.xml"
});

myMenu.addText("text_from", null, "За дата");
myMenu.addInput("date_from", null, "", 75);
// get inputs

```



```

input_from = myMenu.getInput("date_from");
input_from.setAttribute("readOnly", "true");
// init calendar
myCalendar = new dhtmlXCalendarObject([input_from]);
myCalendar.setDateFormat("%Y-%m-%d");
myCalendar.attachEvent("onHide", function () {
    if (myMenu.getValue("date_from") != "") {
        myGrid.filterBy(1, function (a) {
            return (a == myMenu.getValue("date_from"));
        });
    }
});

myMenu.attachEvent("onClick", function (id) {
    if (id == "delete")
    {
        var selection = myGrid.getSelectedRowId();
        myGrid.cells(selection, 1)
        var valueCell = myGrid.cells(selection, 0).getValue();
        deleteTask(valueCell);
        updateGrids();
    }
});

```

```
        clearMyFormView();

    }

    if (id == "reload") {
        updateGrids();
    }

    if (id == "filterUrgent")
        myGrid.filterBy(4, function (a) {
            return (a == "Да");
        });

    if (id == "filterOptional")
        myGrid.filterBy(4, function (a) {
            return (a == "He");
        });

    if (id == "filterFinished")
        myGrid.filterBy(5, function (a) {
            return (a == "Да");
        });

    if (id == "filterUnfinished")
        myGrid.filterBy(5, function (a) {
            return (a == "He");
        });

});
```

```

myGrid = myLayoutManage.cells("a").attachGrid();
myGrid.setImagePath("imgs/");
    <?php include('getTasks.php'); ?>
myGrid.enableAlterCss("even", "uneven");
myGrid.loadXML("xml/text.xml");
myGrid.setEditable(true);
myGrid.attachEvent("onRowSelect", doOnRowSelected)

myFormView = myLayoutManage.cells("b").attachForm();
myFormView.loadStruct("json/formView.json");
myFormView.attachEvent("onButtonClick", function (id) {
    if (id == "save") {
        myFormView.send("updateTask.php", "post", function (loader, response) {
            alert(response);
            updateGrids();
            clearMyFormView();
        });
    }
})

//ДИАГРАМА НА ЗАДАЧИТЕ

```

```

myLayoutTaskStat = mySidebar.cells("day").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Диаграма"}, {id: "b", text: "Диапазон от дни"}]
});

myFormStatTasks = myLayoutTaskStat.cells("b").attachForm();
myFormStatTasks.loadStruct("json/formStatTasks.json");
myFormStatTasks.attachEvent("onButtonClick", function (id) {
    if (id == "go") {
        alert('sending');
        myFormStatTasks.send("getStatTasks.php", "post", function (loader,
            response) {
                alert(response);
                attachChartTasks();
            });
    }
})

//ДОВАБИ ПРИХОД/РАЗХОД

```

```

myFormPayment = mySidebar.cells("addInOut").attachForm();
myFormPayment.loadStruct("json/formPayment.json");
myFormPayment.attachEvent("onButtonClick", function (id) {
    if (id == "send") {
        myFormPayment.send("savePayment.php", "post", function (loader, response) {
            alert(response);
            updateGrids();
        });
        clearMyFormPayment();
    }
});

```

//УПРАВЛЕНИЕ НА ПЛАЩАНИЯ

```

myLayoutPayments = mySidebar.cells("managePayments").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Приходи и разходи", height: "190"}, {id: "b", text:
    "Преглед"}]
});

```

```

myGridPayments = myLayoutPayments.cells("a").attachGrid();
myGridPayments.setImagePath("imgs/");
<?php include('getPayments.php'); ?>
myGridPayments.enableAlterCss("even", "uneven");
myGridPayments.loadXML("xml/gridPayments.xml");
myGridPayments.setEditable(true);

myMenuPayments = mySidebar.cells("managePayments").attachToolbar({
    icons_path: "common/imgs/",
    xml: "xml/toolbarPayments.xml"
});

myMenuPayments.addText("text_from", null, "За дата");
myMenuPayments.addInput("date_from", null, "", 75);
// get inputs
input_from1 = myMenuPayments.getInput("date_from");
input_from1.setAttribute("readOnly", "true");
// init calendar
myCalendarPayments = new dhtmlXCalendarObject([input_from1]);
myCalendarPayments.setDateFormat("%Y-%m-%d");
myCalendarPayments.attachEvent("onHide", function () {
    if (myMenuPayments.getValue("date_from") != "") {
        myGridPayments.filterBy(2, function (a) {
            return (a == myMenuPayments.getValue("date_from"));
        });
    }
});

```

```

        });
    }
});

myMenuPayments.attachEvent("onClick", function (id) {
    if (id == "delete")
    {
        var selectionPayments = myGridPayments.getSelectedRowId();
        var valueCellPayments = myGridPayments.cells(selectionPayments,
            0).getValue();
        deletePayment(valueCellPayments);
        myGridPayments.deleteSelectedItem();
        updateGrids();
        clearMyFormViewPayments();
    }

    if (id == "reload")

        updateGrids();

    if (id == "incomes")
        myGridPayments.filterBy(1, function (a) {
            return (a == "Приход");
        });
    });
});

```

```

        });
    if (id == "outcomes")
        myGridPayments.filterBy(1, function (a) {
            return (a == "Разход");
        });
    if (id == "filterFinished")
        myGridPayments.filterBy(5, function (a) {
            return (a == "Да");
        });
    if (id == "filterUnfinished")
        myGridPayments.filterBy(5, function (a) {
            return (a == "Не");
        });
});

myFormViewPayments = myLayoutPayments.cells("b").attachForm();
myFormViewPayments.loadStruct("json/formViewPayments.json");
myFormViewPayments.attachEvent("onButtonClick", function (id) {
    if (id == "save") {
        myFormViewPayments.send("updatePayments.php", "post", function (loader,
            response) {
                alert(response);
                updateGrids();
            }
        );
    }
});

```



```
        clearMyFormViewPayments();
    });
}
})
```

```
myGridPayments.attachEvent("onRowSelect", doOnRowSelectedPayments)
```

```
//ДИАГРАМА НА ПЛАЩАНИЯТА
```

```
myLayoutPaymentStat = mySidebar.cells("calculate").attachLayout({
    pattern: "2E",
    cells: [{id: "a", text: "Диаграма"}, {id: "b", text: "Диапазон от дни"}]
});
```

```
myFormStatPayments = myLayoutPaymentStat.cells("b").attachForm();
myFormStatPayments.loadStruct("json/formStatPayments.json");
myFormStatPayments.attachEvent("onButtonClick", function (id) {
```

```

        if (id == "go") {
            alert('sending');
            myFormStatPayments.send("getStatPayments.php", "post", function (loader,
            response) {
                alert(response);
                attachChartPayments();
            });
        }
    })

```

//НАСТРОЙКИ

```

myFormSettingsB =mySidebar.cells("settings").attachForm();
myFormSettingsB.loadStruct("json/formSettingsB.json");
myFormSettingsB.attachEvent("onButtonClick", function (id) {
    if (id == "send") {
        myFormSettingsB.send("updateSettings.php", "post", function (loader,
        response) {
            alert(response);
            clearMyFormSettings();
        }
    }
});

```

```

        location.reload();

    });

    }

});

    }

});

}

function doOnRowSelected(id) {

    var valueCellTitle = myGrid.cells(id, 2).getValue();
    myFormView.setItemValue("title", valueCellTitle);

    var valueCellDesc = myGrid.cells(id, 3).getValue();
    myFormView.setItemValue("description", valueCellDesc);

    var valueCellDate = myGrid.cells(id, 1).getValue();
    if (valueCellDate != "Безсрочно")

```

```

        myFormView.setItemValue("date", valueCellDate);
    else
        myFormView.setItemValue("date", "");

    if (myGrid.cells(id, 4).getValue() == "Да") {
        myFormView.setItemValue("urgent", true);
    }

    var valueCellId = myGrid.cells(id, 0).getValue();
    myFormView.setItemValue("id", valueCellId);

    var completed;
    if (myGrid.cells(id, 5).getValue() == "Да")
        completed = 1;
    else
        completed = 0;
    myFormView.setItemValue("completed", completed);

    myFormView.disableItem("id");

}

function doOnRowSelectedPayments(id) {

```

```
var valueCellDescPayments = myGridPayments.cells(id, 3).getValue();
myFormViewPayments.setItemValue("description", valueCellDescPayments);

var valueCellDate = myGridPayments.cells(id, 2).getValue();
if (valueCellDate != "Безсрочно")
    myFormViewPayments.setItemValue("date", valueCellDate);
else
    myFormViewPayments.setItemValue("date", "");

var valueCellAmountPayments = myGridPayments.cells(id, 4).getValue();
myFormViewPayments.setItemValue("amount", valueCellAmountPayments);

var urgent;
if (myGridPayments.cells(id, 5).getValue() == "Да")
    urgent = 1;
else
    urgent = 0;
myFormViewPayments.setItemValue("urgent", urgent);

var type;
if (myGridPayments.cells(id, 1).getValue() == "Приход")
    type = "1";
else
```

```

        type = "0";
myFormViewPayments.setItemValue("type", type);

var completed;
if (myGridPayments.cells(id, 6).getValue() == "Да")
    completed = 1;
else
    completed = 0;
myFormViewPayments.setItemValue("completed", completed);

var valueCellIdPayments = myGridPayments.cells(id, 0).getValue();
myFormViewPayments.setItemValue("id", valueCellIdPayments);

myFormViewPayments.disableItem("id");
}

function doOnRowSelectedTodayPayments() {
    var selectionTodayPayments = myGridTodayPayments.getSelectedRowId();
    var valueCellPayments = myGridTodayPayments.cells(selectionTodayPayments, 1).getValue();
    var myMenuTodayPaymentsItem;
    if (valueCellPayments == "Приход")

```

```

        myMenuTodayPaymentsItem = "Отбележи като получен";
    else
        myMenuTodayPaymentsItem = "Отбележи като платен";
    myMenuTodayPayments.setItemText("mark", myMenuTodayPaymentsItem);
}

function deletePayment(id)
{
    $.ajax({
        type: 'POST',
        data: {"paymentId": id},
        url: 'deletePayment.php',
        success: function (response) {
            alert(response);
            $("#return").text("result from php " + response);
        }
    });
}

function deleteTask(id)
{
    $.ajax({
        type: 'POST',
        data: {"taskId": id},

```

```

        url: 'deleteTask.php',
        success: function (response) {
            alert(response);
            $("#return").text("result from php " + response);
            updateGrids();
        }
    });

}

function completeTask(id)
{
    $.ajax({
        type: 'POST',
        data: {"taskId": id},
        url: 'setTaskCompleted.php',
        success: function (response) {
            alert(response);
            $("#return").text("result from php " + response);
            updateGrids();
        }
    });
}

```



```

function completePayment(id)
{
    $.ajax({
        type: 'POST',
        data: {"paymentId": id},
        url: 'setPaymentCompleted.php',
        success: function (response) {
            alert(response);
            $("#return").text("result from php " + response);
            updateGrids();
        }
    });
}

```

```

function attachChartTasks() {
    myChartTasks = myLayoutTaskStat.cells("a").attachChart({

view: "bar",
    container: "chartDiv",
    value: "#taskscount#",
    gradient: "falling",
    color: "#b9a8f9",

```

```
        radius: 0,  
        alpha: 0.5,  
        border: true,  
        width: 70,  
        xAxis: {  
            template: "#date#",  
            title: "Задачи по дни"  
        },  
        yAxis: {  
            start: 0,  
            end: 10,  
            step: 1,  
        }  
    });  
myChartTasks.load("xml/chartTasks.xml");  
myChartTasks.sort({  
    by: "#date#",  
    dir: "asc",  
    as: "string"  
});
```

```

}

function attachChartPayments() {
    myChartPayments = myLayoutPaymentStat.cells("a").attachChart({
        view: "bar",
        container: "chartDiv",
        value: "#amount#",
        gradient: "falling",
        color: "#b9a8f9",
        radius: 0,
        alpha: 0.5,
        border: true,
        width: 70,
        xAxis: {
            template: "#date#",
            title: "Платания в суми"
        },
        yAxis: {
            start: 0,
            end: 1000,
            step: 100,
        }
    });
    myChartPayments.load("xml/chartPayments.xml");
}

```

```
        myChartPayments.sort({
            by: "#date#",
            dir: "asc",
            as: "string"
        });
    }

function updateGrids()
{

    myGridPayments.loadXML("xml/gridPayments.xml");
    myGridTodayPayments.loadXML("xml/todayPayments.xml");
    myGrid.loadXML("xml/text.xml");
    myGridTodayTasks.loadXML("xml/todayTasks.xml");
}

function clearmyForm()
{
    myForm.setItemValue("id", "");
    myForm.setItemValue("title", "");
    myForm.setItemValue("description", "");
    myForm.setItemValue("date", "");
}
```

```
        myForm.setItemValue("name", "");
        myForm.setItemValue("urgent", false);
    }
function clearMyFormView()
{
    myFormView.setItemValue("id", "");
    myFormView.setItemValue("title", "");
    myFormView.setItemValue("description", "");
    myFormView.setItemValue("date", "");
    myFormView.setItemValue("name", "");
    myFormView.setItemValue("urgent", false);
    myFormView.setItemValue("completed", false);
}

function clearMyFormViewPayments()
{
    myFormViewPayments.setItemValue("id", "0");
    myFormViewPayments.setItemValue("type", "1");
    myFormViewPayments.setItemValue("description", "");
    myFormViewPayments.setItemValue("date", "");
    myFormViewPayments.setItemValue("amount", "");
    myFormViewPayments.setItemValue("name", "");
    myFormViewPayments.setItemValue("urgent", false);
}
```

```
        myFormViewPayments.setItemValue("completed", false);
    }

function clearMyFormPayment()
{
    myFormPayment.setItemValue("id", "0");
    myFormPayment.setItemValue("type", "1");
    myFormPayment.setItemValue("description", "");
    myFormPayment.setItemValue("date", "");
    myFormPayment.setItemValue("amount", "");
    myFormPayment.setItemValue("name", "");
    myFormPayment.setItemValue("urgent", false);
    myFormPayment.setItemValue("completed", false);
}

function clearMyFormSettings()
{
    myFormSettingsB.setItemValue("background", "");
    myFormSettingsB.setItemValue("even", "");
    myFormSettingsB.setItemValue("uneven", "");
}
```

```
</script>

<meta http-equiv="Cache-Control" content="no-cache, no-store, must-revalidate" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="0" />
</head>

<body onload="doOnLoad();" bgcolor= <?php echo $_SESSION['background'] ?> >
<?php include('menu.php') ?>
<b> &nbsp; &nbsp; &nbsp; Здравейте, <?php echo $_SESSION['userfirst'] . " " . $_SESSION['userlast'] ?>!
</b>
    <div id="sidebarObj">
        <form action="userPage.php" method="post" target="[some target]">
            <div id="myForm"></div>
        </form>
    </div>

</body>
</html>
```

