****

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА**

**Факултет по Изчислителна Техника и Автоматизация**

**КАТЕДРА „Софтуерни и интернет технологии“**

**„Разработка на програмни модули за съставяне на Учебен разпис –**

**Uni-Scheduler“**

**Дипломна работа за придобиване на образователно-квалификационна степен „бакалавър“**

**Дипломант: Научен ръководител:**

Николай Цветанов Георгиев / х.доц. М. Митев /

Факултетен № 61562162

Специалност: Софтуерни и Интернет Технологии

**Вapнa**

**2019**

[**„Разработка на програмни модули за съставяне на Учебен разпис –** 1](#_Toc18206151)

[**Uni-Scheduler“** 1](#_Toc18206152)

[Глава 1. Увод 4](#_Toc18206153)

[Глава 2. Обзор и анализ на алгоритмите и програмите за разработка програмни модули за съставяне на учебен разпис 5](#_Toc18206154)

[2.1 Анализ на подобен тип системи 5](#_Toc18206155)

[2.1.1 Система за изготвяне на учебен график “StudyGizmo” 5](#_Toc18206156)

[2.1.2 Система за изготвяне на учебен график “Schedule My Teachers” 7](#_Toc18206157)

[2.1.3 Система за изготвяне на учебен график “ Schedule Builder Online” 8](#_Toc18206158)

[2.2 Цели 9](#_Toc18206159)

[2.2 Таргет група от хора, към които е фокусирано приложението. 10](#_Toc18206160)

[2.3 Общи изисквания на потребителя 12](#_Toc18206161)

[2.3 Общи изисквания на потребителя 12](#_Toc18206162)

[2.4 Use-case 12](#_Toc18206163)

[2.4.1 Преподавател 12](#_Toc18206164)

[2.4.3 Администратор (Разпореждащия учебни занятия) 13](#_Toc18206165)

[2.5 Прототип на Uni-scheduler 14](#_Toc18206166)

[Глава 3: Технологична схема за разработка на учебни разписи 17](#_Toc18206167)

[3.1 Описание на данните 17](#_Toc18206168)

[3.1.1 Представяне на роли 17](#_Toc18206169)

[3.1.4 Представяне на бизнес правила 18](#_Toc18206170)

[3.1.5 Определяне на обектите с техните атрибути и релациите 20](#_Toc18206171)

[3.1.6 Релационна диаграма 27](#_Toc18206172)

[3.2 Използвани технологии 28](#_Toc18206173)

[3.2.1 Уеб приложение 28](#_Toc18206174)

[3.2.1.1 Клиент-сървър 28](#_Toc18206175)

[3.2.1.2 Клиент 29](#_Toc18206176)

[3.2.1.3 Сървър 29](#_Toc18206177)

[3.2.2 Структура на уеб приложението 30](#_Toc18206178)

[3.2.2.2 Обектно-релационно картографиране(ORM) 32](#_Toc18206179)

[3.2.2.3 Технологична рамка(framework) 33](#_Toc18206180)

[3.2.2.4 Виртуални машини: 34](#_Toc18206181)

[3.2.2.4 Vagrant: 35](#_Toc18206182)

[3.2.2.5 Ansible: 35](#_Toc18206183)

[3.2.2.6 Secure Shell (SSH): 35](#_Toc18206184)

[3.2.2.7 Система за контрол на версиите (Git): 36](#_Toc18206185)

[3.2.2.8 „S.O.L.I.D” принципи: 36](#_Toc18206186)

[3.2.2.9 PHP (Hypertext Preprocessor) 37](#_Toc18206187)

[3.2.2.10 „Symfony“ (Технологична рамка) 38](#_Toc18206188)

# Глава 1. Увод

В наши дни хората почти не могат да си представят живота си без невероятните технологии и чудните устройства, които само преди десетилетия се срещаха единствено на страниците на научнофантастичните романи. Компютри, смартфони, таблети, лаптопи – всеки от нас, ежедневно използва тези „плодове на научно-техническия прогрес". Всички тези устройства са направени с цел улеснение на човешките дейности. Както ни дават възможността да извършваме сложни и времеемки задачи, свързани със изчисление или извличане на информация, така и осигуравя до голяма степен достоверност или автентичност на извършената с помоща на тези технологии, задача. Друга услуга която ни придоставят е способността за експресното разгласяването на важна информация.

Един страхотен пример за нуждата от създаване на технология, която предостовя иммено бързина, автентичност и достъп на информацията е тази която дава възможност за разпореждане на задълженията на един преподавател във висшите учебни заведения. В някои от тях, тази дейност се извършва от хора, които нямат в наличие, подобна технология и единствения начин за справяне с такава задача е прибягването към старите методи за състаяване, разгласяне и осигуряването на автентичност, което е именно изписването им на хартиен носител. Това само по себе си води до допускането на грешки, което пък води до още по-голяма загуба на време, поради факта, че поправянето на една грешка в един график може да доведе до много други проблеми, също така предоставянето на информацията е отново чрез разгласяване на многобройни листи от хартия, които в последвствие ще трябва да се подновят при разрешаване на проблема. Прост пример за това са колизиите в графиците на преподавателите между упражнения които се водят в една и съща стая по едно и също време. Избягване на такъв тип проблеми възникнали поради човешка грешка при съставянето на графика за един преподавател не само влияе на самите преподаватели но и на студентите които губят ценно време от занятието, което би трябвало в този момент да имат. За осъществяването на тази задачата, чиято цел е да състави учебните разписи на всички преподаватели във висшето учебно заведение, за кратък период от време, са нужни доста хора. При наличието на такава система, броят на тези хора ще бъде сведен до един човек.

# Глава 2. Обзор и анализ на алгоритмите и програмите за разработка програмни модули за съставяне на учебен разпис

## 2.1 Анализ на подобен тип системи

### C:\Users\nikol\Desktop\screencapture-freecollegeschedulemaker-2019-08-27-21_02_02.png2.1.1 Система за изготвяне на учебен график “StudyGizmo”

Фиг.1 – Начална странца на уебсайта за система създаване на графици “StudyGizmo“.

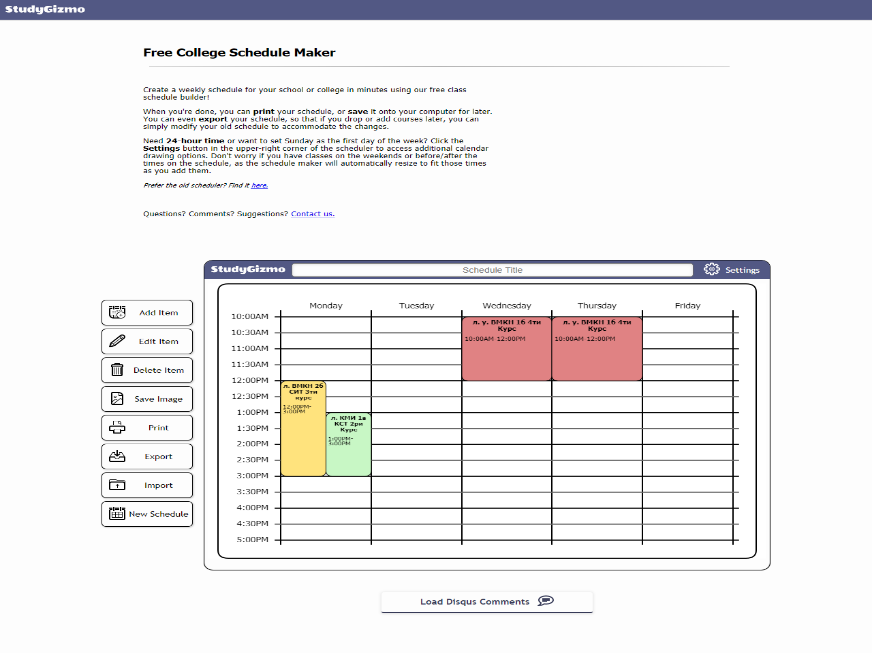
Една от доста срещаните системи, разработени с цел, съставянето на учебни разписание е иммено уеб платформата „StudyGizmo“. Тази система е доста често използвана, не само от преподаватели, но и от студенти. Достъпна е от всякакви съвременни устройства които имат достъп до интернет мрежа, като, смартфони, лаптопи, компютри и таблети. Простия дизайн на системата предоставя възможостта на потребителите да съставят без да изпитват затруднения, графици, които са лесно разчитими.

**Плюсове**:

* Опростен и интуитивен дизайн за бързо и лесно ползване.
* Безплатна
* Достъпна
* Даваща възможност за създаване, редактиране или изтриване на занятия или събития
* Позволява запазване на изготвения график в формат на снимка
* Позволява въвеждането на вече съществуващи графици

**Минуси:**

* Системата не предоставя възможността от изготвянето на комплексни графици.
* Системата е несъвместима с повечето мобилни устройства
* Липсват алгоритми, осигуряващи автентичност на графиците

Фиг. 2 – Примерен график, изготвен от преподавател

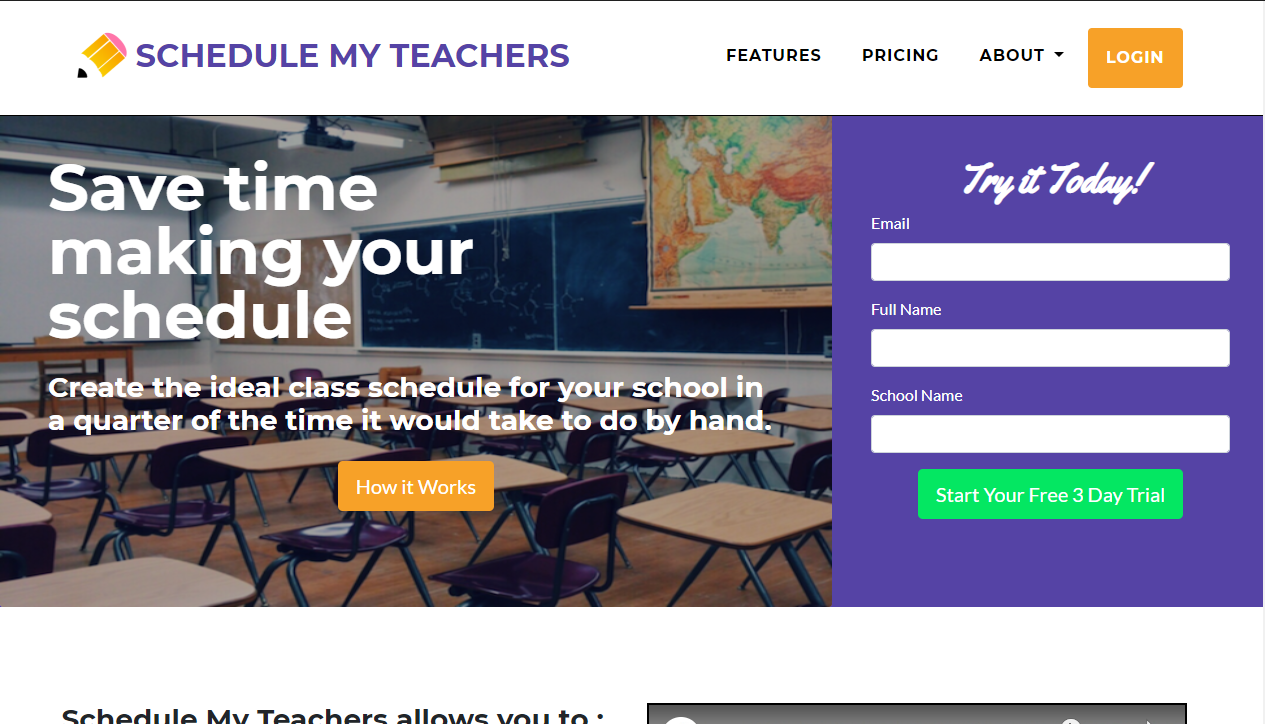
### 2.1.2 Система за изготвяне на учебен график “Schedule My Teachers”

Друга доста разпространена сред учебните заведения, система, която за разликата от “StudyGizmo”. Целта на приложението е да съставя графици на занятията за преподаватели и ученици. Дава възможност за

**Плюсове:**

* Съставяне на множество от графици за учители в дадено учебно заведение
* Автоматично генериране на график
* Изготвяне на график по свой вкус, в дни удобни за теб.
* Равномерно разпределяне на занятията през седмицата.
* Опростен и интуитивен дизайн за бързо и лесно ползване.

**Минуси:**

* Системата не е достъпна за външни лица.
* Услугата е платена
* Не е предназначена за висшите училищни заведения.
* Не предоставя възможност за смяна на езика на работа

Фиг. 3 – Начална страница на “Schedule My Teachers”

### 2.1.3 Система за изготвяне на учебен график “ Schedule Builder Online”

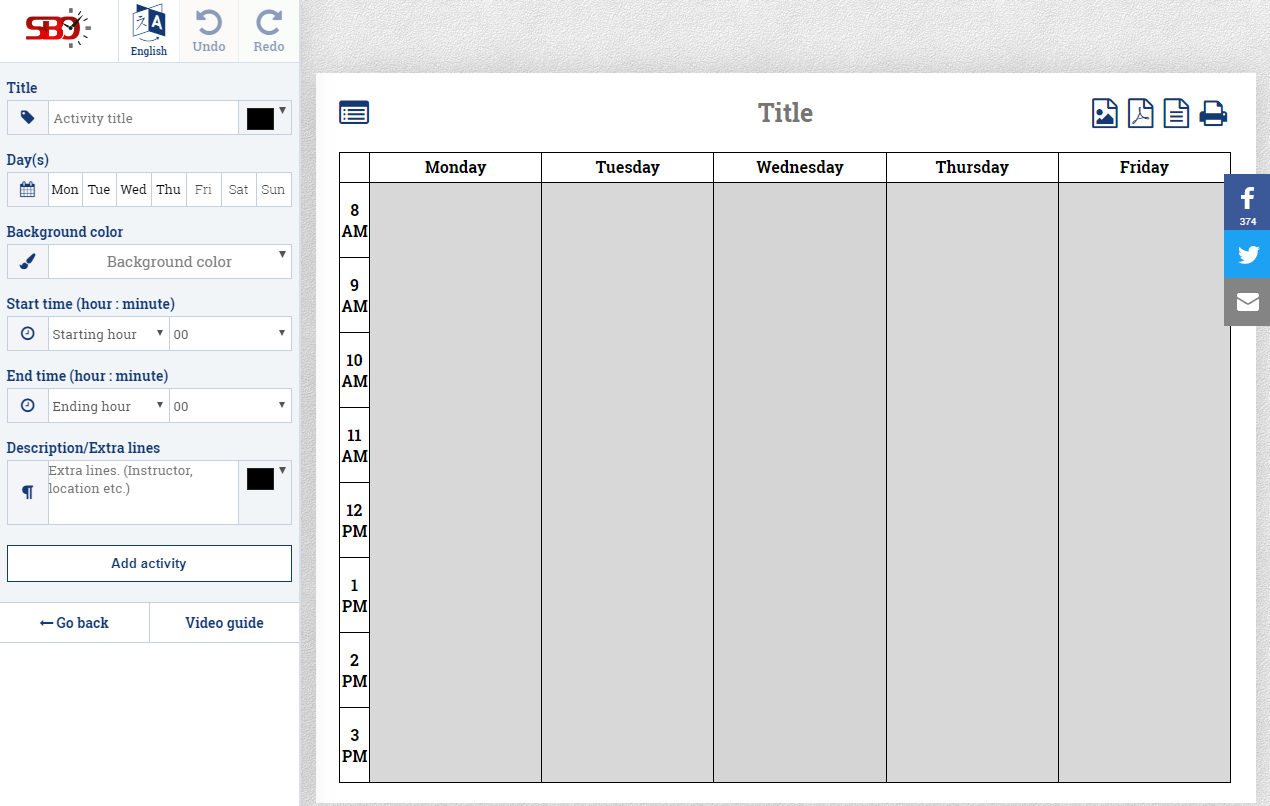
**“Schedule Builder Online”** не е много популярна система за изготвяне на учебни графици. Сравнение със другите подобни на тази технология за съставяне на разписания системи, “Scheduler Builder Online” предлага не малко допълнителни услуги като:

* Запазване на изготвения график под формата на: jpg(графично изображение), pdf(преносим формат на документа) или разпечатване на хартиен носител при наличието на принтер от страна на клиента.
* Създаване на безплатен профил, позволяващ общуване между регистрираните потребители чрез вградена чат система.
* Допълнително „разкрасяване“ на изготвения график като например задаването на фон на изготвения график.

**Минуси:**

* Не толкова удобен за ползване, но пък е предоставено ръководство за употреба.
* Според потребителите, предоставели обратна връзка, свързана със функционалността на системата, се разбира, че има доста технически дефекти.
* Системата не е приспособена за работа от смартфони, таблети или устройства с по-малък екран.

Фиг. 4 – Начална страница на **“Schedule Builder Online”**



**2.1.4 Извод от анализа на представените системи за изготвяне на учебен график**

Според разгледаните по-горе приложения, става ясно, че повечето безплатни за употреба, разработени системи за съставяне на учебни графици са:

* Не завършени
* Не добре оптимизирани от гледна точка на дизайн
* За разработката на системите са използвани доста остарели версии на езици за програмиране, които водят до технически дефекти и възможни пробиви във сигурността.
* Липсващ алгоритъм за автентичност на графиците, който следи за конфликти между занятията или събитията, които биват разпоредени в даден ден и час.
* Липсващ алгоритъм за следене на дублиране на занятия или събития, въведени от потребителите

Но от друга страна, те ни придоставят доста услуги, които сами по себе си са достатъчни за изготвянето на макар и индивидуален, функциониращ учебен график. Подобни безплатни типове системи като “Schedule Builder Online” и „StudyGizmo“ са достъпни и до голяма степен биха свършили работа ако става на въпрос за изготвянето на сравнително прост, индивидуален учебен седмичен график. Друга положителна страна е гъвкавостта им. Те могат да бъдат ползвани както за учебни занятия, така и за най различни събития от ежедневието.

Възможни и подобрения на анализираните системи:

* Интегриране на алгоритъм за автентичност на графиците.
* Подобряване графичния интерфейс
* Разширяване на системите с цел даване възможност на потребителите да използват приложенията от мобилни устройства като смартфони и таблети

## 2.2 Цели

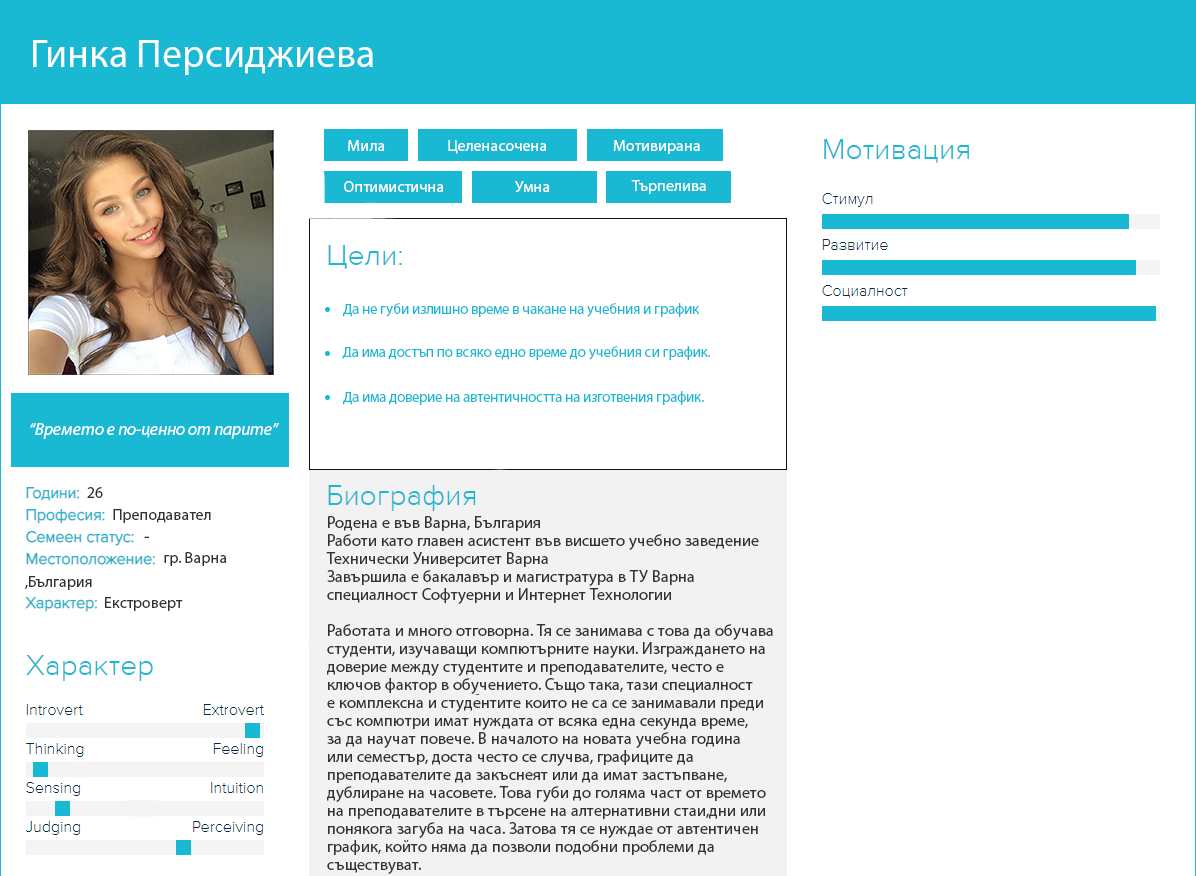
Целта на разработката е създаване на система, която ще предоставя възможност на персонала, отговорен за изготвянето на учебните разписания за преподавателите във висше учебно заведение, да изготвят за кратък период от време, автентични графици на учебните занятията. Едно от най-важните функционалности на системата е проверката за автентичност на даден график. Блогадарение на която, не позволява на персонала, изготвящ дадено разписание да допускат грешки при създаването. Системата не само ще намали до голяма степен времето за изготвяне на разписание, но и ще намали заетоста на персонала, на който ще му бъде възложена тази задача. Всеки изготвен график ще бъде публикуван на уеб платформа, която е достъпна от устройства, ползващи интернет, като смартфони, таблети, лаптопи, компютри. Тази платформа е достъпна единствено от оторезирани преподаватели. Всеки един преподавател има създаден профил, в който се намират неговите разпоредени от оторизиран персонал на висшето учебно заведение, занятия. Той, както и други регистрирани във системата преподаватели, имат достъп до информацията помежду си. Благодарение на скороста и автентичноста на изготвените графици, преподавателите ще избегнат проблеми, като застъпване на стаи, упражнения, дублиране, закъснение при разгласяване на разписанията и други възникнали през годините проблеми.

Системата предлага:

* Опростен и интуитивен дизайн за бързо и лесно ползване от страна на преподавателите
* Опростен и интуитивен дизайн за бързо и лесно ползване от страна на персонала, отговорен за изготвянето на учебния разпис.
* Възможност за достъп от преподавателите до издадените графици по всяко едно време, без значение от устройството, стига то да бъде свързано с интернет мрежата.
* Автентичност, изразяваща се във алгоритъм за валидиране при създаване на графиците
* Моментална актуализация при промяна на график
* Възможност за работа разстояние, касаеща изготвящия графици персонал

## 2.2 Таргет група от хора, към които е фокусирано приложението.

Приложението е насочено за хората, част от висшите учебни заведения, като студенти, преподаватели и персонала, който се занимава със изготвянето на учебните разписания. Разработената система е предназначена само и единствено от този оторизирани потребители. Хора които не са част от висшето учебно заведение нямат достъп до тази функционалност.

Фигура.5 - Потребител, който е част от таргета, към който е фокусирана платформата.

Фигура 5 е един пример за потребител, който вероятно ще се нуждае от услугите, които платформата предлага или с чиято цел е създадена. Накратко това е потребител, чиято работа е да бъде преподавател във висше учебно заведение. Занимава се със обучението на студенти и най-важното нещо за него е да не губи времето на студентите си. Често се случва, в разписанието за часовоте да има грешки или да се получават конфликти с разписанията на други преподаватели, което довежда в загубата на времето на преподавателя и студентите които имат занятие.

## 2.3 Общи изисквания на потребителя

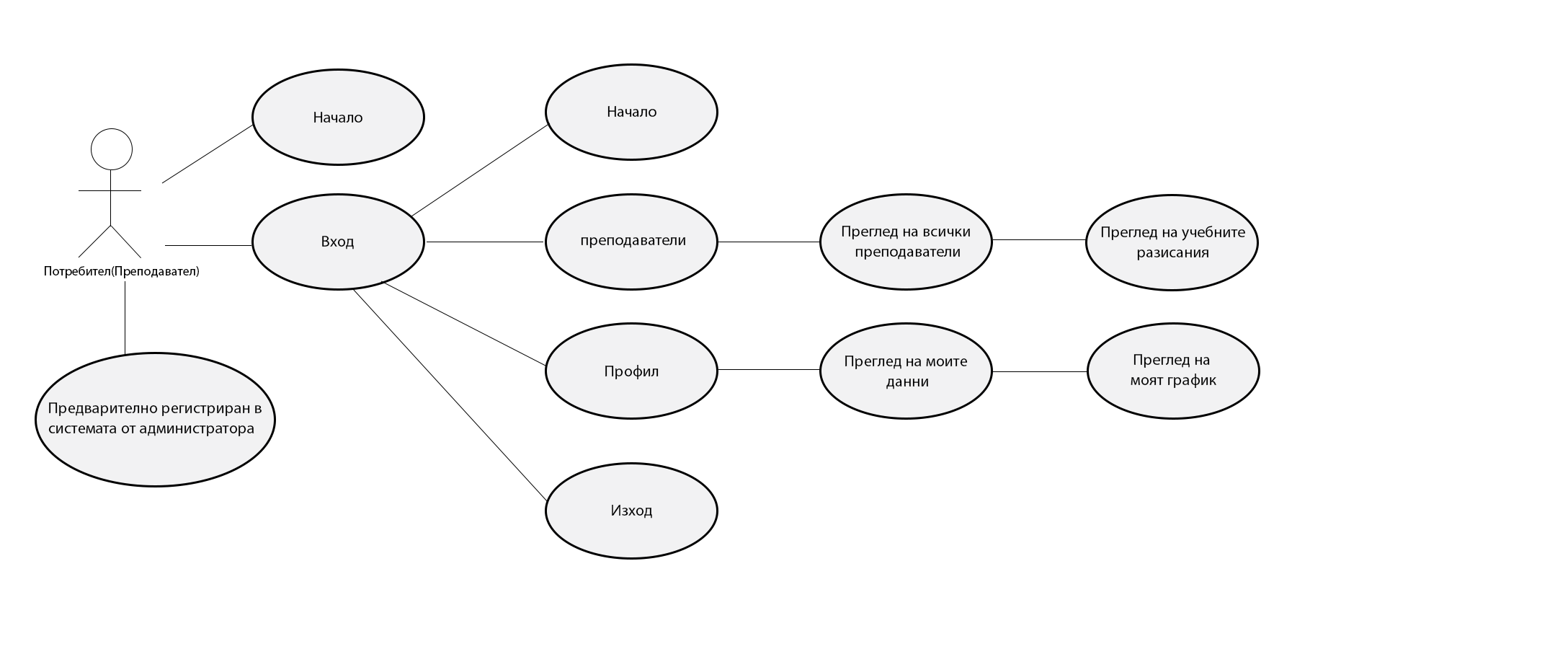
* Опростен и интуитевен дизайн за бързо и лесно ползване.
* Постигане на женалия резултат с възможно най-малко стъпки.
* Платформата да е лека и оптимизирана
* Данните на всеки един потребител да бъдат защитени.
* Уеб платформата да покрива всички защитни механизми с цел защита на информацията на потребителите

## 2.3 Общи изисквания на потребителя

* Автентичност на учебните разписания
* Лесен достъп до графиците
* В случай на промяна. Възможността за моментална проверка на новостите в разписанието

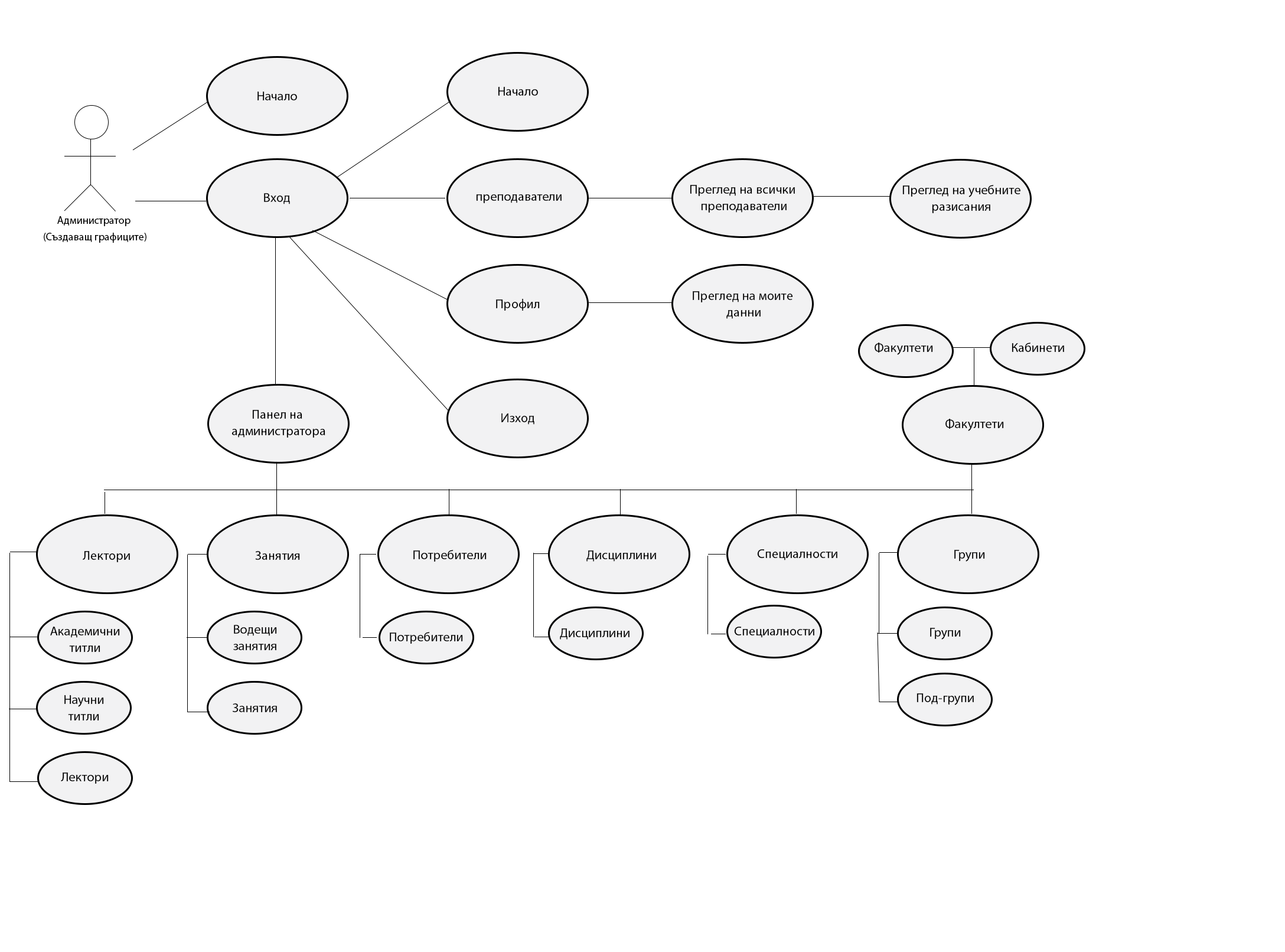
## 2.4 Use-case

### 2.4.1 Преподавател



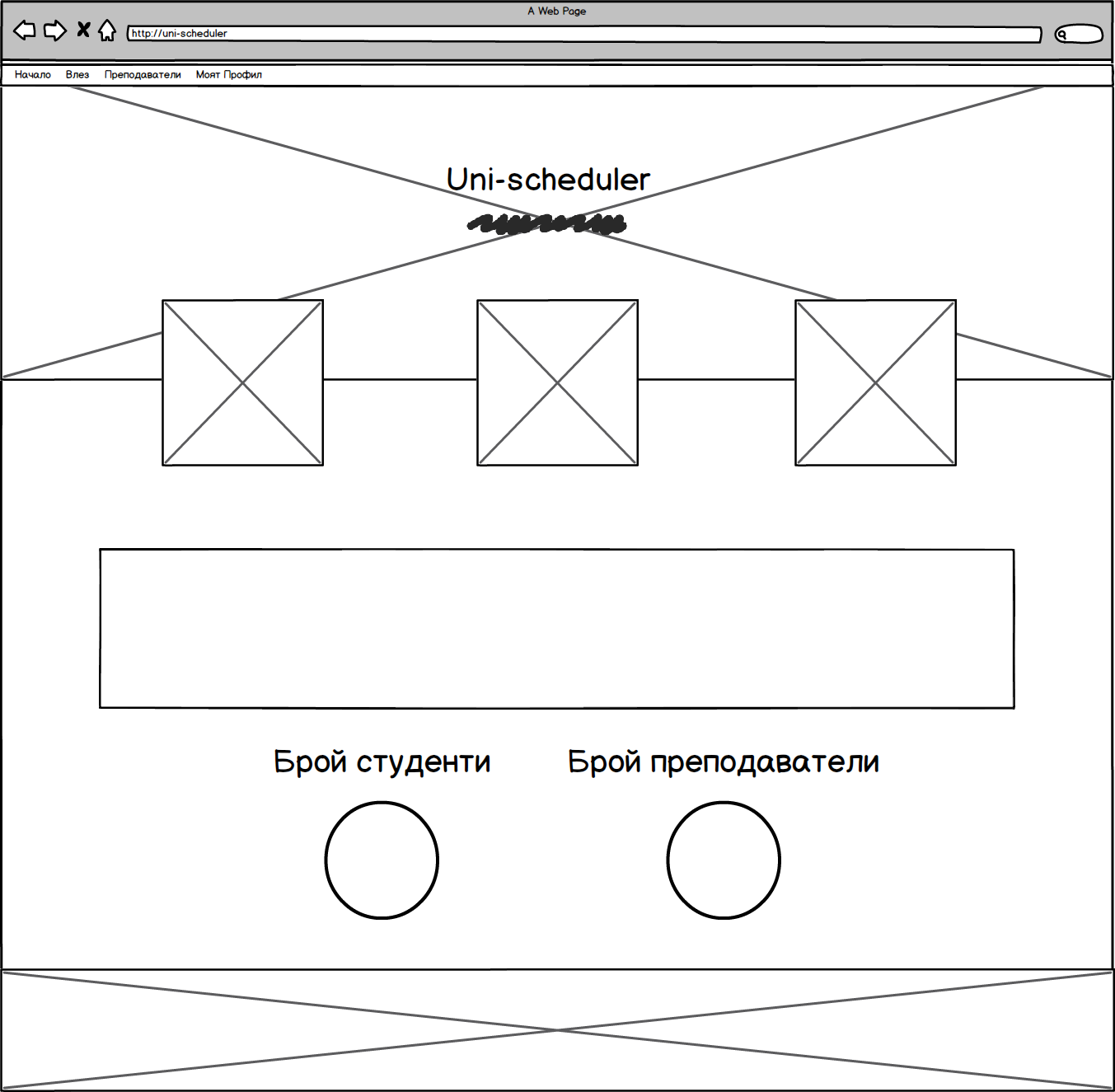
Фиг. 6 – Use-case диаграма за преподавателите.

### 2.4.3 Администратор (Разпореждащия учебни занятия)

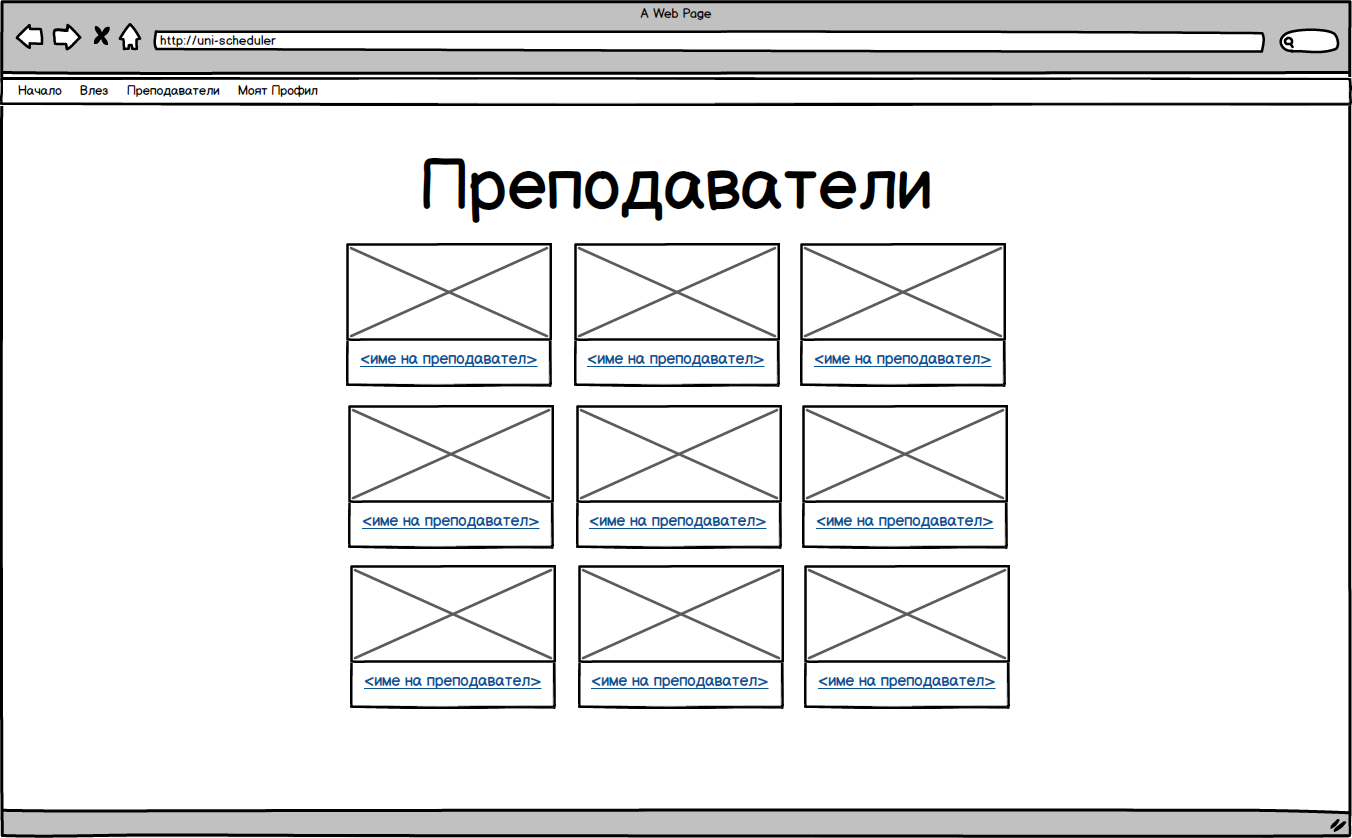


Фиг. 6 – Use-case диаграма за администратор.

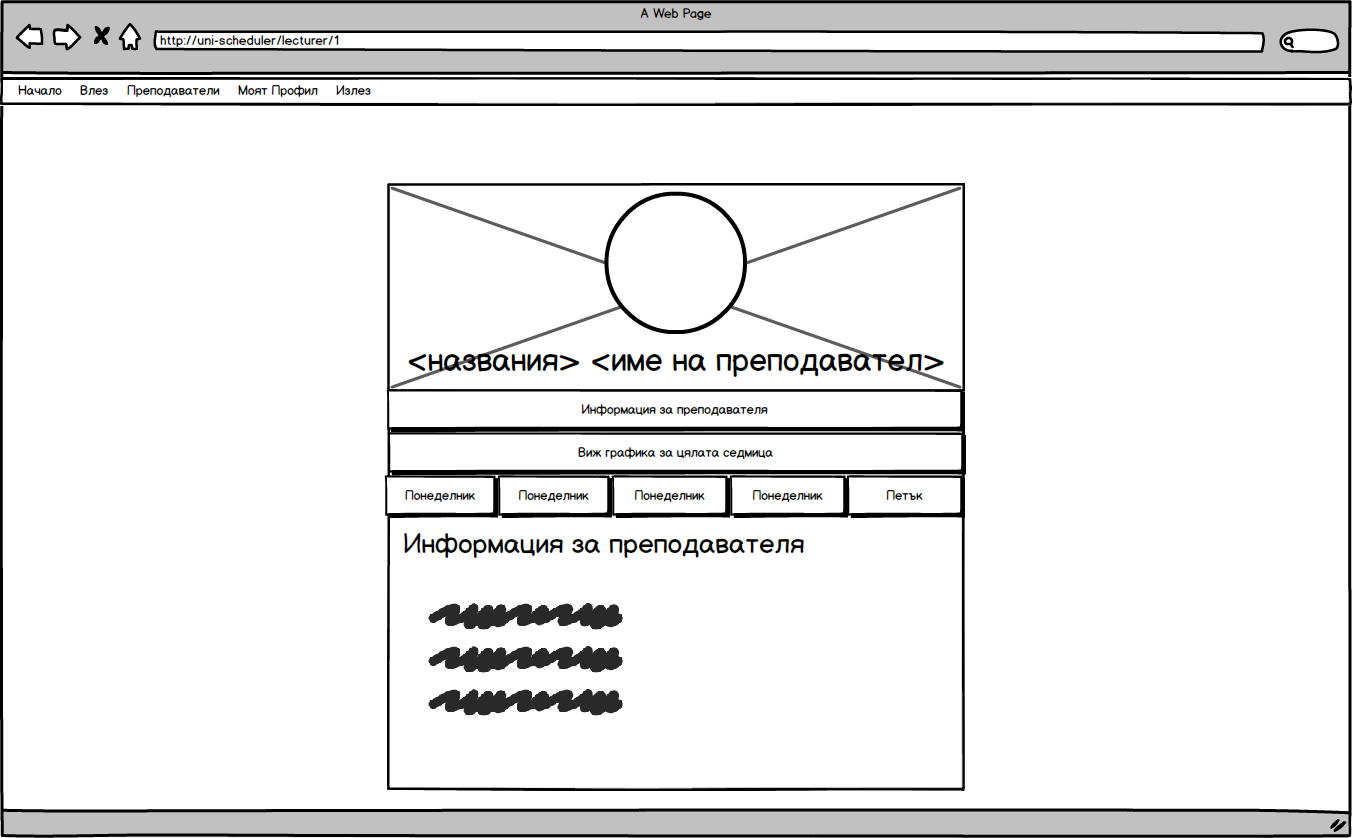
## 2.5 Прототип на Uni-scheduler

Фиг. 7 Протип на начална страница

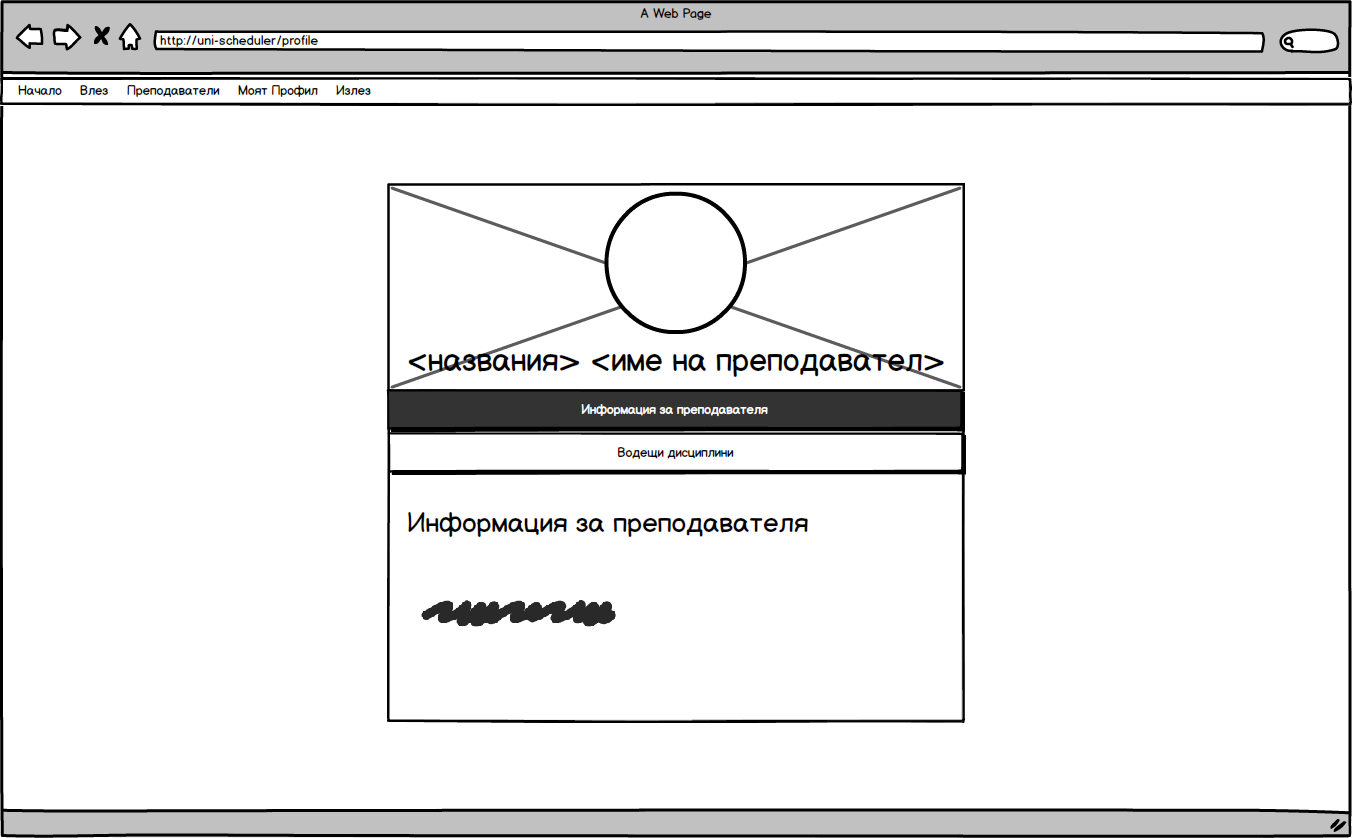
Фиг. 8 Прототип на страница за преглед на преподавателите



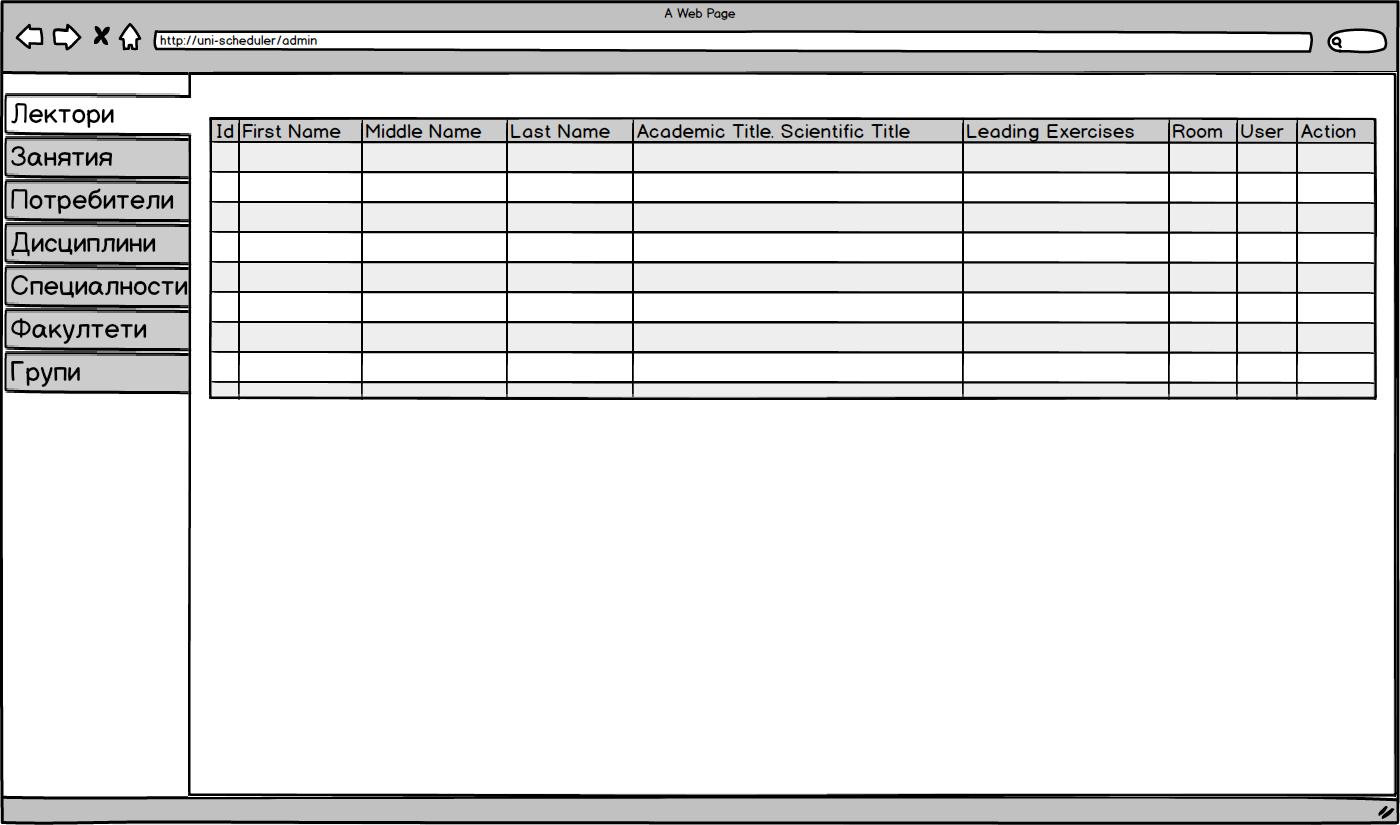
Фиг. 9 Прототип на страницата за даден преподавател



Фиг. 10 Прототип на страницата за информация на потребителя



Фиг. 11 Протип на администраторският панел



# **Глава 3: Технологична схема за разработка на учебни разписи**

## 3.1 Описание на данните

### 3.1.1 Представяне на роли

Потребителска роля е предварително зададена категория, които могат да бъдат присвоявани на потребителите въз основа на техните длъжност или други критерии. Роли обикновено се използват за представяне на персонализирани версии или изгледи на шаблон на формуляр към различни типове потребители. Има много сценарии, където потребителските роли са от полза за потребители. Например с потребителски роли, може:

* Показвате уникален изглед за всяка потребителска роля
* Показвате различна секция за всяка потребителска роля
* Задаване на проверка на данни за поле, който се базира на всяка потребителска роля
* Подаване на данни на формуляр към външен източник на данни, който се базира на всяка потребителска роля

**Преподавател**

Потребителите с роля „Преподавател“ имат достъп до списък със всички преподаватели и до техните учебни разписи. Всеки преподавател може да разгледа своите данни, както и поетите учебни занятия. Потребителите с тази роля нямат достъп до администраторският панел, който позволява да се съдават, изтриват и корегират информация.

**Администратор**

Потребителите с роля „Админстратор“ имат възможността да ползват всички услуги които един преподавател може да използва. Този потребител има достъп до администраторския панел. Работата на администратора е да създава, редактира и изтрива информация. Той се занимава със разпореждането на занятията. Има достъп до информацията на всички регистрирани преподаватели. Има правото да възлага администраторски привилегии на потребители.

### 3.1.4 Представяне на бизнес правила

Бизнес правилата са много важна част от изграждането на базата данни. Според тях се създават обектите с техните специфични атрибути, връзките между самите обекти и валидациите(ограничения). Има два вида бизнес правила – структурни и процедурни.

* Структурните – определят информацията която се съхранява.
* Процедурни – представляват изискванията на работния процес на даден бизнес.

**Структурни**

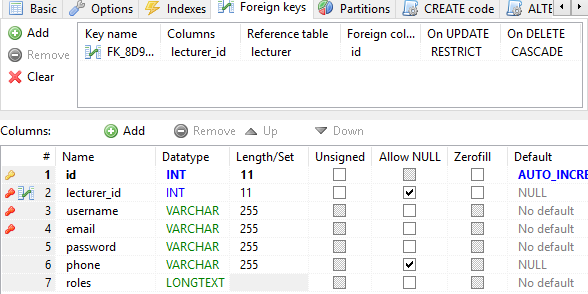
* Един преподавател може да има само един потребителски акаунт
* Един потребителски акаунт може да бъде използван само от един преподавател
* Един преподавател може да има само една академична титла
* Много преподаватели може да имат много научни титли
* Един преподавател може да има много водещи занятия
* Много водещи занятия могат да принадлежат само на една специалност
* Много водещи занятия могат да принадлежат само на една дисциплина
* Едно водещо занятие може да има много занятия
* Една група може да има в себе си много под-групи
* Много групи могат да имат само една специалност
* Много под-групи могат да имат само една група
* Един факултет може да съдържа в себе си много стаи
* Много стаи могат да бъдат съдържани в един факултет
* Много занятия могат да имат една стая
* Много занятия могат да имат едно водещо занятие
* Много занятия могат да имат една под-група

**Процедурни**

* Потребители могат да се впишат в системата, използвайки имейл и парола.
* Преподавателите могат да разглеждат други преподаватели и техните учебни разписания
* Администратор може да разпорежда занятия на преподавател, ако този преподавател е рапореден да води на съответната дисциплина
* Администратор разпоредил занятие за даден преподавател, може да го изтрие или поправи
* Търсене на занятия по конкретен преподавател, начало на занятие и ден с цел проверка за автентичност
* Търсене на водещо занятие по конкретен преподавател, конкретна дисциплина и тип на водещото занятие с цел избягване на дублиращи се разпоредени водещи занятия
* Търсене на водещо занятие по конкретна дисциплина и тип на водещото занятие с цел избягване от разпореждане на водещо занятие от тип лекция, на повече от един преподавател
* Търсене на дисциплина по конкретно име и абревиатура с цел избягване на дублиращи се дисциплини
* Търсене на занятие по начало на занятието, водещо занятие, ден, край на занятието, стая и под-група с цел избягване на вече поети/дублиращи се занятия
* Търсене на факултети по име и абревиатура с цел избягване на дублиращи се факултети
* Търсене на групи по име с цел избягване създаване на групи, които вече съществуват за съответната специалност
* Търсене на стаи по номер на стая и факултет с цел избягване от дублиране на стаи които вече принадлежат на съответният факултет.
* Търсене на специалност по начало на година(поток), име и абревиатура с цел избягване дублиране на специалности с еднакви имена и година(поток) които вече съществуват.
* Търсене на под-групи по име, принадлежаща на съответна група с цел избягване дублиране на вече съществуваща под-група.
* Търсене на потребители по потребителско име, имейл и преподавател с цел избягване дублиране на потребители
* Търсене не преподаватели с цел визуализиране броя на преподавателите
* Търсене на конкретен преподавател по конкретен ден от седмицата с цел визуализация на занятията поети в този ден и този преподавател

### 3.1.5 Определяне на обектите с техните атрибути и релациите

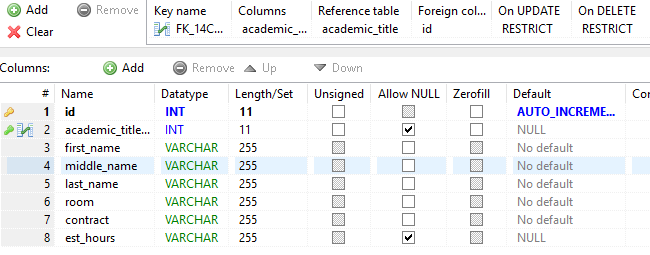
**Потребители (User)**



Фиг. 12 – Таблица User, която съдържа информация за потребителите.

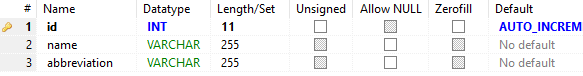
* Идентификационен номер(id) – първичен ключ
* Вторичен клюз за преподавател(lecturer\_id) свързан с първичния ключ към таблица на преподавателите, уникално поле
* Потребителско име(username) – уникално и задължително поле
* Имей (email) – уникално и задължително
* Парола (password) – криптирана с „Bcrypt“ алгоритъм
* Телефонен номер (phone) – незадължително поле
* Роля (role) – задължително поле

**Преподавател (Lecturer)**



Фиг. 13 Таблица Lecturer, която съдържа информация за преподавателите

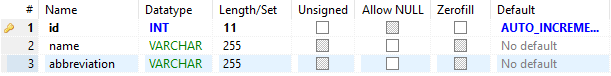
* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за академично название на преподавателя, свързан с таблица „academic\_title“
* Име (first\_name) – задължително поле
* През име(middle\_name) - задължително поле
* Фамилия(last\_name) - задължително поле
* Стая на преподавателя (room) – задължително поле
* Тип на договора (contract) - задължително поле



**Академично название (academic\_title)**

Фиг. 14 Таблица „academic\_title”, която съдържа видовете академични названия

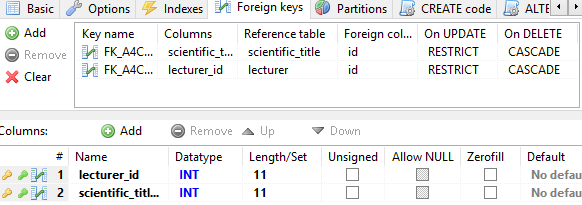
* Идентификационен номер(id) – първичен ключ
* Име (name) – задължително поле
* Абревиатура (abbreviation) – задължително поле

**Научно название (scientific\_title)**

Фиг. 14 Таблица „scientific\_title“, която съдържа информация за всички научни названия

* Идентификационен номер(id) – първичен ключ
* Име (name) – задължително поле
* Абревиатура (abbreviation) – задължително поле

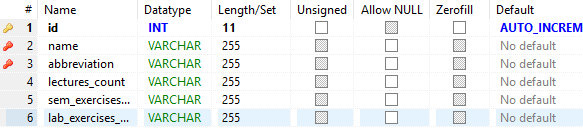
**Научно название / Преподаватели (lecturer\_scientific\_title)**



Фиг. 15 Таблица „lecturer\_scientific\_title“, която играе ролята на междинна таблица, свързваща таблицата на преподавателите и научните названия

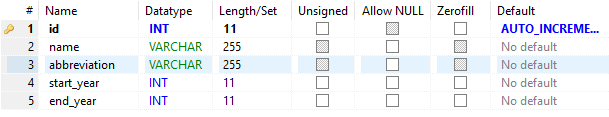
* Вторичен ключ за научно название, свързан с таблица „scientific\_title“
* Вторичен ключ за преподавател, свързан с таблица „lecturer“

**Дисциплини (discipline)**



Фиг. 16. Таблица „discipline”, която съдържа информация за дисциплините

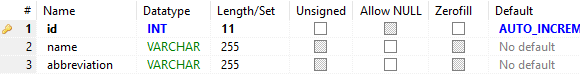
* Идентификационен номер(id) – първичен ключ
* Име (name) – задължително поле
* Абревиатура (abbreviation) – задължително поле
* Брой лекции за семестъра (lectures\_count) – задължително поле
* Брой семинарни упражнения за семестъра (sem\_exercise\_count) – задължително поле
* Брой лабораторни упражнения за семестъра (lab\_exercise\_count) – задължително поле

**Специалност (specialty)**

Фиг. 17. Таблица „specialty”, която съдържа информация за специалностите.

* Идентификационен номер(id) – първичен ключ
* Име (name) – задължително поле
* Абревиатура (abbreviation) – задължително поле
* Първа година на специалността/поток (start\_year) – задължително поле
* Последна година на специалността/поток (end\_year) – задължително поле

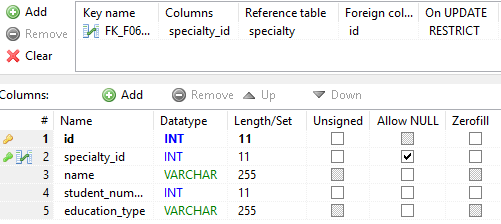
**Факултет (faculty)**



Фиг. 18 Таблица „faculty ”, която съдържа информация за факултетите.

* Идентификационен номер(id) – първичен ключ
* Име (name) – задължително поле
* Абревиатура (abbreviation) – задължително поле

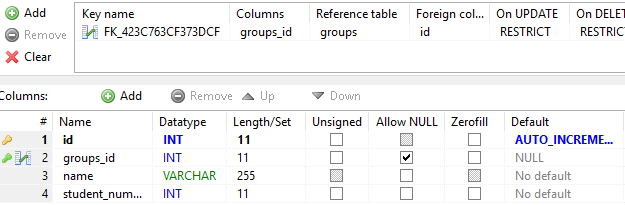
**Групи (groups)**



Фиг. 19 Таблица „groups ”, която съдържа информация за групите.

* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за специалност, свързан с таблица „specialty“
* Име (name) – задължително поле
* Брой студенти (student\_number) – задължително поле
* Тип на обучение (education\_type) – задължително поле

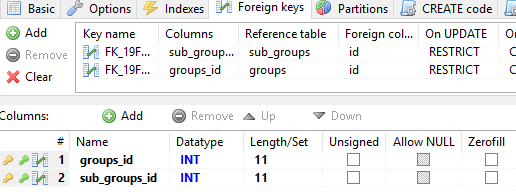
**Под-групи (sub\_groups)**



Фиг. 20 Таблица „sub\_groups ”, която съдържа информация за под-групите.

* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за група, свързан с таблица „groups“
* Име (name) – задължително поле
* Брой студенти (student\_number) – задължително поле

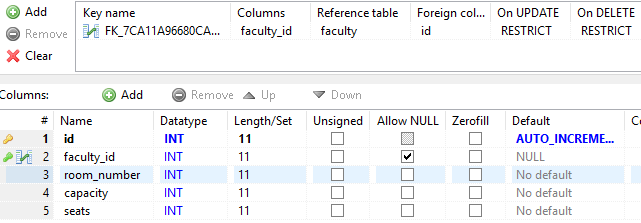
**Групи / Под-групи (sub\_groups)**



Фиг. 21 Таблица „groups\_sub\_groups”, която играе ролята на междинна таблица, свързваща таблицата „groups” и „sub\_groups“

* Вторичен ключ за група, свързан с таблица „groups“
* Вторичен ключ за под-група, свързан с таблица „sub\_groups“

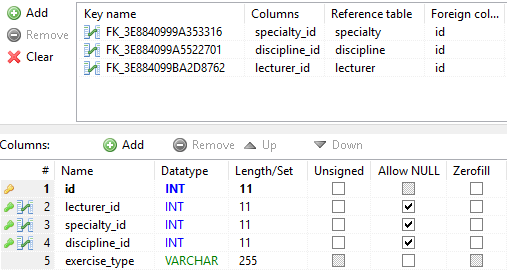
**Стаи (rooms)**



Фиг. 22 Таблица „rooms”, която съдържа информация за стаите.

* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за факултет, свързан с таблица „faculty“
* Номер на стая (room\_number) – задължително поле
* Капацитет (capacity) – задължително поле
* Работни места (seats) – задължително поле

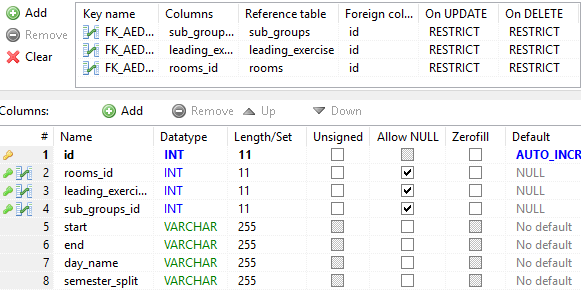
**Водещи занятия (leading\_exercises)**



Фиг. 23 Таблица „leading\_exercises”, която съдържа информация за водещите занятия.

* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за преподавател, свързан с таблица „lecturer“
* Вторичен ключ за специалност, свързан с таблица „specialty“
* Вторичен ключ за дисциплина, свързан с таблица „discipline“
* Вид на занятието – задължително поле

**Занятия (exercises)**

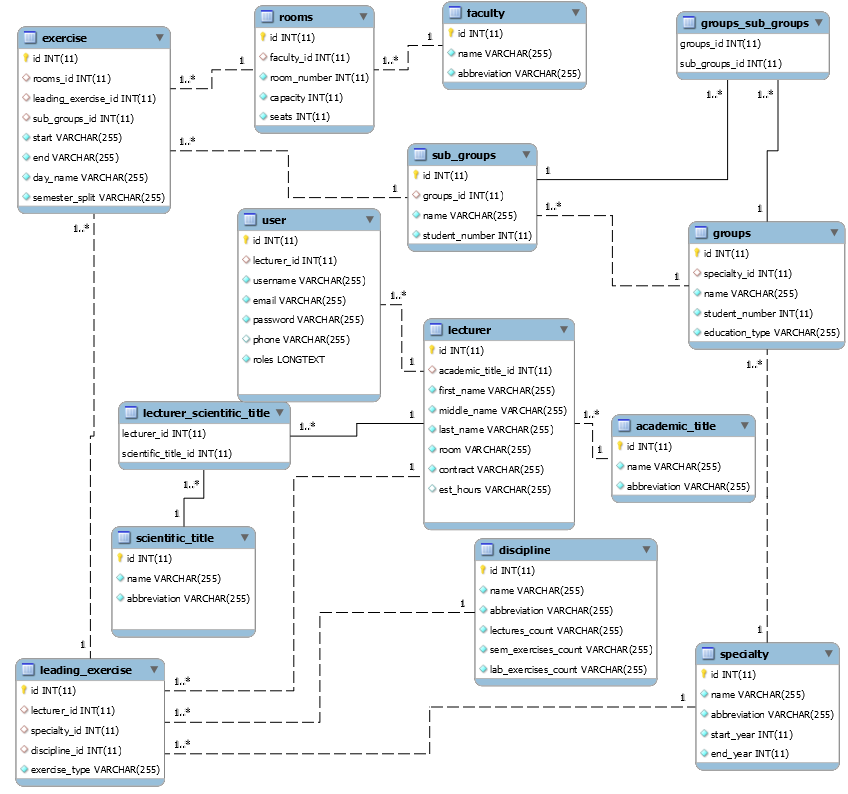


Фиг. 24 Таблица „exercises”, която съдържа информация за занятията.

* Идентификационен номер(id) – първичен ключ
* Вторичен ключ за стаи, свързан с таблица „rooms“
* Вторичен ключ за водещи занятия, свързани с таблица „leading\_exercises“
* Вторичен ключ за под-групи, свързани с таблица „sub-groups“
* Начало на занятието – задължително поле
* Край на занятието – задължително поле
* Име на ден от седмицата – задължително поле
* Зимен/Летен семестър – задължително поле

### 3.1.6 Релационна диаграма

Фиг. 25 – Релационна диаграма



## 3.2 Използвани технологии

### 3.2.1 Уеб приложение

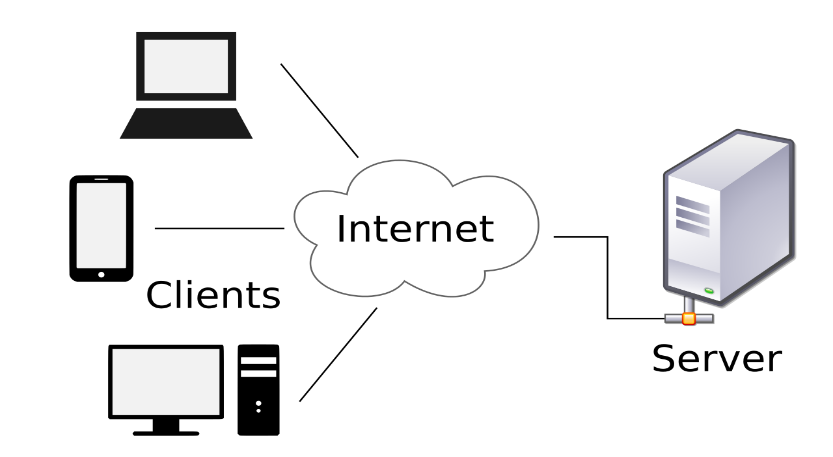
Уеб приложение (Уеб приложение) е приложна програма, която се съхранява на отдалечен сървър и се достъпва през Интернет чрез браузърен интерфейс. Уеб услугите са уеб приложения по дефиниция и много, макар и не всички, уебсайтове съдържат уеб приложения. Те могат да бъдат проектирани за голямо разнообразие и могат да бъдат използвани от организация до физическо лице по множество причини. Често използваните уеб приложения могат да включват уеб поща, онлайн калкулатори или магазини за електронна търговия. Някои уеб приложения могат да бъдат достъпни само от конкретен браузър; повечето обаче са достъпни независимо от браузъра.

**Ползи:**

Уеб приложенията имат много различни приложения, а с тези приложения има много потенциални ползи. Някои техни общи предимства включват:

* Разрешаване на достъп на множество потребители до една и съща версия на приложение.
* Не е необходимо да се инсталират уеб приложения.
* Достъпа до уеб приложенията може да се осъществява чрез различни платформи като десктоп, лаптоп или мобилни.
* Достъпът може да се осъществява чрез множество браузъри.

### 3.2.1.1 Клиент-сървър

Това е архитектура на компютърна мрежа, в която много клиенти (отдалечени процесори) искат и получават услуга от централизиран сървър (хост компютър). Клиентските компютри предоставят интерфейс, който позволява на компютърния потребител да изисква услуги на сървъра и да Фиг. 26 – Клиент-сървър архитектура

показва резултатите, които сървърът връща. Сървърите изчакват заявките да пристигнат от клиентите и след това отговарят на тях. В идеалния случай сървърът предоставя стандартизиран прозрачен интерфейс за клиентите, така че те не трябва да са запознати със спецификата на системата (т.е. хардуера и софтуера), която предоставя услугата. Често са разположени на работни станции или на персонални компютри, докато сървърите са разположени другаде в мрежата, обикновено на по-мощни машини. Този изчислителен модел е особено ефективен, когато клиентите и сървърът имат различни задачи, които рутинно изпълняват. Например в болничната обработка на данни клиентският компютър може да работи с приложна програма за въвеждане на информация за пациента, докато сървърният компютър работи с друга програма, която управлява базата данни, в която информацията е трайно съхранявана. Много клиенти могат да получат достъп до информацията на сървъра едновременно и в същото време клиентският компютър може да изпълнява други задачи, като изпращане на електронна поща.

### 3.2.1.2 Клиент

Клиентът може да бъде обикновено приложение или цяла система, която получава достъп до услуги, предоставяни от сървър. Клиентът може да се свърже със сървър чрез различни средства като домейни сокети, имена, споделена памет или чрез интернет протоколи, което е най-разпространеният метод, използван след широкото приемане на Интернет.

### 3.2.1.3 Сървър

Почти всички персонални компютри могат да служат като мрежови сървъри. Въпреки това обикновено компютрите със софтуерна / хардуерна система имат функции и конфигурации само за тази задача. Например, специализираните сървъри може да имат високоефективна „RAM“ памет, по-бърз процесор и няколко твърди диска с голям капацитет. В допълнение, специализирани сървъри могат да бъдат свързани към излишни източници на захранване, няколко мрежи и други сървъри. Такива функции и конфигурации на връзката са необходими, тъй като много клиентски машини и клиентски програми могат да зависят от тях, за да функционират ефективно, правилно и надеждно.

За да работи в уникалната мрежова среда, в която много компютри и хардуер / софтуерни системи зависят само от един или няколко сървърни компютъра, сървърът често има специални характеристики и възможности, включително:

* Възможността за актуализиране на хардуер и софтуер без рестартиране.
* Разширена възможност за архивиране на критични данни.
* Напреднала производителност в мрежата.
* Автоматичен (невидим за потребителя) пренос на данни между устройства.
* Висока сигурност на ресурси, защита на данните и паметта.

Сървърните компютри често имат специални операционни системи, които обикновено не се срещат на персонални компютри. Някои операционни системи се предлагат както в сървърна, така и в настолна версия и използват подобни интерфейси.

### 3.2.2 Структура на уеб приложението

**3.2.2.1** Модел-Изглед-Контролер (MVC)

Шаблонът определя не само ролите, които обектите играят в приложението, той определя начина, по който обектите общуват помежду си. Всеки от трите типа обекти е отделен от останалите чрез абстрактни граници и комуникира с обекти от другите типове през тези граници. Колекцията от обекти от определен тип „MVC“ в едно приложение понякога се нарича слой - например моделен слой. MVC е в основата на добрия дизайн за приложение. Ползите от възприемането на този модел са многобройни. Много обекти в тези приложения са по-често използваеми и техните интерфейси са по-добре дефинирани. Приложенията, използващи този дизайн също са по-лесно разширяващи се от другите приложения. Освен това много технологии и архитектури са базирани на MVC и изискват използването на този шаблон.

**Моделни обекти(Model):**

Тези обекти защитават данните, специфични за дадено приложение и определят логиката и изчисленията, които манипулират и обработват тези данни. Например, моделен обект може да представлява герой в игра или контакт в адресната книга.Той може да има връзки към един и към много други подобни обекти и затова понякога моделния слой на приложението ефективно е една или повече обектни графики. Голяма част от данните, които са част от устойчивото състояние на приложението, трябва да пребивават в моделните обекти след зареждането на данните в приложението. В идеалния случай обектът на модела не трябва да има изрична връзка с обектите на изглед (View), които представят неговите данни и позволяват на потребителите да редактират тези данни - не трябва да се занимават с потребителски интерфейс и проблеми с представянето.

* **Комуникация:**

Действията на потребителя в изгледания слой, които създават или променят данни, се предават чрез контролер и водят до създаването или актуализирането на моделен обект. Когато моделен обект се промени (например нови данни се получават през мрежова връзка), той уведомява контролера, който актуализира съответните изгледи.

**Изгледни обект(View):**

Oбект в приложение, което потребителите могат да виждат. Той знае как да визуализира себе си и може да реагира на действията на потребителя. Основна цел на обектите за изглед е да се показват данни от моделните обекти на приложението и да се даде възможност за редактиране на тези данни. Въпреки това, обектите на изглед обикновено се отделят от моделни обекти в MVC приложение.

* **Комуникация:**

Изгледите научават, че данните в модела са се променили чрез контролер обектите. Всяка една проямяна от потребителя в едно приложение бива съобщавано на котролер, който се грижи за отразяване на информацията в модел обекта

**Контролер обект:**

Контролера действа като посредник между един или повече изгледи на приложение и един или повече от неговите контролер обекти. По този начин те игрят ролята на проводник, чрез който изгледите научават за промените в моделните обекти и обратно. Обектите на контролера също могат да изпълняват задачи за настройка и координиране на приложение и да управляват жизнените цикли на други обекти.

* **Комуникация:**

Контролера интерпретира действията на потребителя, направени в изгледите и съобщава новите или променени данни на моделния слой. Когато обектите на модела се променят, контролера съобщава тези нови данни на изгледите, така че те да могат да визуализират тези промени.

### 3.2.2.2 Обектно-релационно картографиране(ORM)

В компютърните науки означава техника на програмиране за преобразуване на данни между несъвместими типове системи с помощта на обектно-ориентирани езици за програмиране. Това всъщност създава "виртуална база данни на обекти", която може да се използва от езика на програмиране.

В обектно-ориентираното програмиране, задачите за управление на данни действат върху обекти, които почти винаги са не скаларни стойности. Например запис в адресната книга, който представлява един човек заедно с нула или повече телефонни номера и нула или повече адреси. Това би могло да бъде моделирано в обектно-ориентирана реализация от обект „Личност“ с атрибути, за да съдържа всеки елемент от данни, който вписването съдържа: името на лицето, списък с телефонни номера и списък с адреси. Самият списък с телефонни номера би съдържал обект „PhoneNumber“ и т.н. Различни методи могат да бъдат свързани с обекта, като метод за връщане на предпочитания телефонен номер, домашния адрес и т.н.

Въпреки това много популярни продукти на базата данни като системи за управление на бази данни „SQL“ (СУБД) могат да съхраняват и манипулират само скаларни стойности като цели числа и низове, организирани в таблици. Програмистът трябва или да преобразува обектните стойности в групи от по-прости стойности за съхранение в базата данни (и да ги преобразува обратно при извличане), или да използва само прости скаларни стойности в рамките на програмата. Обектно-релационното картографиране реализира първия подход

**Doctrine (ORM)**

„Doctrine2“ е обектно-релационен картограф (ORM) за PHP 7.1+, който осигурява прозрачна устойчивост за PHP обекти. Той цели пълно отделяне на бизнес логиката от постоянството в системата за управление на релационни бази данни. Ползата от „Doctrine“ за разработчика е възможността да се съсредоточи върху обектно-ориентираната бизнес логика.

### 3.2.2.3 Технологична рамка(framework)

Платформа за разработване на софтуерни приложения. Тя осигурява основа, на която разработчиците на софтуер могат да изграждат програми за конкретна платформа. Например, една рамка може да включва предварително създадени класове и функции, които могат да се използват за обработка на вход, управление на хардуерни устройства и взаимодействие със системния софтуер. Това опростява процеса на разработка, тъй като програмистите не трябва да преоткриват колелото всеки път, когато разработят ново приложение.

Технологичната рамка е подобна на интерфейс за приложно програмиране (API), макар че технически рамката включва API. Както подсказва името, една рамка служи като основа за програмиране, докато API осигурява достъп до поддържаните от рамката елементи. Една рамка може също да включва кодови библиотеки, компилатор и други програми, използвани в процеса на разработка на софтуер.

### 3.2.2.4 Виртуални машини:

Виртуална машина (VM) е софтуерна програма или операционна система, която не само показва поведението на отделен компютър, но също така е способна да изпълнява приложения и програми като отделен компютър. Виртуалната машина, обикновено известна като гост, се създава в друга изчислителна среда, наречена "хост". Няколко виртуални машини могат да съществуват в рамките на един хост наведнъж.

Виртуалните машини се реализират чрез методи за емулация на софтуер или техники за хардуерна виртуализация. В зависимост от използването им и нивото на кореспонденция с всеки физически компютър, виртуалните машини могат да бъдат разделени на две категории:

* Системни виртуални машини - системна платформа, която поддържа споделянето на физическите ресурси на хост компютъра между множество виртуални машини, всяка от които работи със собствено копие на операционната система. Техниката за виртуализация се осигурява от софтуерен слой, известен като хипервизор, който може да работи или върху чист хардуер, или операционна система.
* Процесни виртуални машини – Създадени са за осигуряване на независима от платформа среда за програмиране, която маскира информацията на базисния хардуер или операционна система и позволява изпълнението на програмата да се извършва по същия начин във всяка дадена платформа.

**Предимства:**

Позволява множество среди на операционната система на един физически компютър без никаква намеса Виртуалните машини са широко достъпни и са лесни за управление и поддръжка. Предлага възможности за осигуряване на приложения и възстановяване след бедствия

**Недостатъци:**

Те не са толкова ефективни като физическия компютър, защото хардуерните ресурси се разпределят по косвен начин. Множество виртуални машини, работещи на една физическа машина, могат да осигурят нестабилна производителност

### 3.2.2.4 Vagrant:

Vagrant е инструмент за изграждане и управление на среди на виртуални машини в един работен процес. С лесен за използване работен процес и съсредоточаване върху автоматизацията, Vagrant понижава времето за настройка на средата за разработка. Осигурява лесна за конфигуриране, възпроизводима и преносима работна среда, изградена на базата на стандартната за индустрията технология и контролирана от единен последователен работен процес, за да помогне за увеличаване на производителността и гъвкавостта при разработката на приложение. С помоща на скриптови езици като „Ansible“ може да се издигат и конфигурират със всички нужни за разработката на съответното приложение, виртуални машини.

### 3.2.2.5 Ansible:

Ansible е инструмент за автоматизация на информационните технологии. Той може да конфигурира системи, да „установи“ софтуер и да организира по-напреднали ИТ задачи, като непрекъснато внедряване или нулеви актуализации за прекъсване.

Основните цели на Ansible са простотата и лекотата на използване. Той също така има силен акцент върху сигурността и надеждността, като включва минимум движещи се части, използване на „OpenSSH“ за транспорт.

### 3.2.2.6 Secure Shell (SSH):

Известен също като „Secure Shell“ или „Secure Socket Shell“, е мрежов протокол, който предоставя на потребителите, особено на системните администратори, сигурен начин за достъп до компютър през незащитена мрежа. SSH също се отнася до пакета от помощни програми, които реализират „SSH“ протокола. „Secure Shell“ осигурява силна автентификация и криптирана комуникация с данни между два компютъра, свързващи се чрез отворена мрежа, като например интернет. „SSH“ се използва широко от мрежовите администратори за дистанционно управление на системи и приложения, което им позволява да влизат в друг компютър по мрежа, да изпълняват команди и да преместват файлове от един компютър на друг. „SSH“ позволява:

* Сигурен отдалечен достъп до мрежови системи или устройства с активиран „SSH“, както за потребители, така и до автоматизирани процеси
* Сигурни и интерактивни сесии за прехвърляне на файлове
* Автоматизирани и защитени прехвърляния на файлове
* Зищитено управление чрез команди към отдалечени устройства или системи
* Зищитено управление на компоненти на мрежовата инфраструктура.

„SSH“ също често се използва в скриптове и друг софтуер, за да се даде възможност на програми и системи за отдалечен и сигурен достъп до данни и други ресурси.

### 3.2.2.7 Система за контрол на версиите (Git):

Git е разпределена система за контрол на версиите за проследяване на промените в изходния код по време на разработката на софтуер. Той е предназначен за координиране на работата между програмистите, но може да се използва за проследяване на промените във всеки набор от файлове. Целите му включват скорост, цялост на данните и поддръжка за разпределени, нелинейни работни процеси. Както при повечето други разпределени системи за контрол на версиите, и за разлика от повечето системи клиент-сървър, всяка Git директория на всеки компютър е пълноценно хранилище с пълна история и пълни възможности за проследяване на версиите, независимо от мрежовия достъп или централен сървър.

### 3.2.2.8 „S.O.L.I.D” принципи:

„SOLID“ е съкращение за пет на брой важни принципа на дизайна, при работа с обектно ориентирано програмиране

* “**S**” Принцип на единна отговорност - при програмирането Принципът на единната отговорност гласи, че всеки модул или клас трябва да носи отговорност за една част от функционалността, осигурена от софтуера.
* „**O**“ Отворен / затворен принцип - софтуерните единици (класове, модули, функции и т.н.) трябва да бъдат отворени за разширения, но затворени за модификация.
* „**L**“ Принцип на заместване на Лисков - той заявява, че обектите в дадена програма трябва да бъдат заменими с екземпляри от техните подтипове, без да променят правилността на тази програма.
* „**I**“ Принцип на разделяне на интерфейса - никой клиент не трябва да бъде принуждаван да зависи от методи, които не използва.
* „**D**“ Принцип на инверсия на зависимостта - модулите от високо ниво не трябва да зависят от модулите с ниско ниво. И двете трябва да зависят от абстракциите. Абстракциите не трябва да зависят от детайлите. Детайлите трябва да зависят от абстракциите.

Прилагайки тези 5 принципа, които правят SOLID съкращението, ние се възползваме от многократна употреба, поддръжка, мащабируемост и лесна тестваема кодова база.

### 3.2.2.9 PHP (Hypertext Preprocessor)

PHP е скриптов език от страна на сървъра, създаден специално за уеб разработка. Може лесно да бъде вграден в HTML файлове, а HTML кодовете могат да бъдат записани и във PHP файл. Нещото, което го разграничава от езика на клиентска страна като HTML е че PHP кодовете се изпълняват на сървъра, докато HTML кодовете се изобразяват директно в браузъра. PHP кодовете първо се изпълняват на сървъра и след това резултатът се връща в браузъра. Единствената информация, която клиентът или браузърът знае, е резултатът, върнат след изпълнение на PHP скрипта на сървъра, а не действителните PHP кодове, присъстващи във PHP файла. Също така, PHP файловете могат да поддържат други клиентски скриптови езици като CSS и JavaScript.

PHP всъщност може да направи всичко, свързано със скриптове от страна на сървъра. Например, PHP може да:

* Получава данни от формуляри,
* Да генерира динамично съдържание на страници,
* Да работи с бази данни,
* Да създава сесии,
* Да изпраща и получава бисквитки,
* да изпраща имейли и т.н.

В PHP също има много хеш функции, за да криптират данните на потребителя, което прави PHP защитен и надежден за използване като скриптов език от страна на сървъра. Това са някои от способностите на PHP, което го прави подходящ за използване като скриптов език от страна на сървъра.

### 3.2.2.10 „Symfony“ (Технологична рамка)

Symfony е PHP рамка с отворен код , който следва „MVC“ архитектура. Използва се за изграждане на високоефективни сложни уеб приложения. По същество опростява работата на разработчиците и им позволява да създават уеб приложения без монотонно и широко кодиране.

Основната идея на „Symfony Framework“ е да премахне забързаното кодиране и да спести време за развитие. Основава се на философията за създаване на софтуер от потребителите за техните собствени нужди. Разработчиците могат да добавят допълнителни модули, докато продуктът расте. Улеснява живота на програмиста с лесно достъпни компоненти на рамката и конфигурация от висок клас. Рамката осигурява:

* Бързо и лесно разработване на приложения – „Symfony“ изразходва по-малко памет, а рамката позволява на потребителите да създават стабилни и високоефективни приложения.
* Комфорт и удобство – позволява на разработчиците, ползващи рамката, да се съсредоточат повече върху основните и разширени функционалности на приложението, а не върху основните функции. Има и „Symfony“ инструменти за разрешаване на кодиращи грешки и проблеми със сигурността.
* Лесна за употреба и гъвкава - дава пълен контрол върху конфигурацията и други критични характеристики на развитието. Рамката е съвместима с други системи от бази данни. Лесният за потребителя интерфейс с ефективна методология помага за създаването на сложни и динамични приложения с голяма гъвкавост на количеството. Допълнителните функционалности могат лесно да се добавят с множество функции. Той е адаптиран към специфичните изисквания на потребителите.
* Надеждна поддръжка
* Възможност за персонализиране – „Symfony“ предлага страхотни персонализирани функции и функционалности за разработчиците и бизнеса
* Възможност за раширяване - Всичко в „Symfony” се представя като пакет. Всеки пакет има уникална функционалност. Позволява преиползване на пакети. Можете да промените или промените всичко, дори сърцевината на системата, без да конфигурирате отново цялата рамка. Позволява добавяне на функционалности.
* Лесно тестване на програмни фрагменти/функционалности - Улеснява тестването с независима библиотека на „PHP Unit“. Той симулира „HTTP“ заявки и проверява изхода без никакъв скрипт, използвайки инструменти за тестване. Той автоматизира функционалното тестване, което спестява значително време и усилия на разработчиците. Symfony предлага и някои наистина добри инструменти за функционално, поведенческо и единично тестване.

### 3.2.2.11 MySQL

MySQL е подкрепена от „Oracle“ система за управление на релационни бази данни с отворен код, базирана на структуриран език на заявките (SQL). „MySQL“ работи на почти всички платформи, включително „Linux“, „UNIX“ и „Windows“.

SQL е език, чиято основна цел е създаване, обработка, извличане и други операции с данни от релационни системи за управление на бази от данни. Освен стандартния SQL, дефиниран от ANSI и ISO, съществуват още много негови разширения и вариации. Почти всички разработчици предоставят различни вариации и разширения на SQL, които се описват като SQL диалекти. Част от тези разширения са със затворен характер – Oracle PL/SQL и други.

SQL е създаден със специфична цел – заявки за данни в релационна база от данни. Като такъв той е декларативен език за програмиране, а не процедурен. Разширения като PL/SQL намаляват разликата добавяйки процедурни елементи и контролни структури.

**Извличане на данни**

SELECT се използва за извличането на редове от една или повече таблици от базата данни. Това е една от най-често използваната команда. При извличането на данни чрез SELECT е необходимо да бъде описан желания резултат, но без операциите, които трябва да се извършат, за да бъде постигнат той. Често срещани ключови думи, които се използват при SELECT заявка са:

* FROM – използва се за да се определи от кои таблици трябва да бъдат извлечени данните.
* WHERE – за описание на определени условия, които трябва да бъдат изпълнени, за да се смята, че конкретния запис трябва да бъде върнат.
* GROUP BY – служи за групиране на резултата от редове по някакви данни.
* ORDER BY – служи за сортиране на резултата от редове в някакъв ред

**Основни ключови думи са CREATE и DROP:**

* CREATE – създава обект (пр. таблица) в базата данни
* DROP – изтрива съществуващ обект от базата данни.

**Обработка на данни**

Има няколко стандартни групи от запазени думи в SQL, като една от тях е Езика за обработка на данни (DML). Той се използва да добавяне, модифициране и изтриването на такива.

* INSERT – добавя запис в съществуваща таблица.
* UPDATE – модифицира данните на вече съществуващ запис.
* MERGE – комбинира данни от множество таблици.
* TRUNCATE – изтрива всички данни от таблица.
* DELETE – премахва записи от таблица.

.