*В**ътрешна система за администриране на работници в фирма-клиент*

Заглавие:

Проектна група **№ 14**

**Автори**

* Алберто Каракутев, фак.№ 471218074
* Георги Барбутов, фак.№ 471218013
* Георги Гинев, фак.№ 471218052
* Симона Кънчева, фак.№ 471218022

**Дата:** 23.10.20 г.

**Съдържание:**

1. **Въведение**
2. **Предназначение:**

* **Обхват на проекта**
* **Актьори**
* **Използвани термини и символи**
* **Източници**

1. **Архитектурен обзор**
2. **Изглед на данни**
3. **Нефункционални изисквания**

Въведение

Вътрешната система за администриране на работници в организация ще бъде с уеб базиран потребителски интерфейс.

Тя ще предоставя възможност за създаване и промяна на графици, въвеждане и преглед на информация за конкретен служител, въвеждане и преглед на документи и вътрешен форум за комуникация между служителите.

**Основни роли в системата:**

1. Администратор
2. Служител

 Предназначение

 Обхват

Уеб-приложение, изградено чрез Java Spring сървърна част и JavaScript  
дизайн на уеб-страниците. Системата ще се занимава с административни дейности и процеси, отнасящи се до персонал на дадена фирма-клиент. Софтуерът е предназначен да предлага обща, схематизирана информация, до която работниците да имат достъп, с цел подобрена комуникация между тях и фирмата. Предоставя се и администраторски достъп, който позволява промени на грешна или невярна информация. Чрез имплементиране на MVC архитектура, изградена от модулни микропроцеси с цел разширяемост и надежност, се осигурява свобода при специфиране на поръчка на софтуера от фирмата-клиент. Стремежа е предоставяне на базов, мултифункционален софтуер, който дава възможност за лесно и надежно разширяване при конкретна заявка. Софтуерът ще предоставя 5 основни модула.

* **Работнически график** – Изчисляване, вписване и модифициране на графика за дадено време в бъдещето на конкретен работник. Чрез определени нива на работниците, те ще имат достъп до графиците на своите колеги от същото или по-ниско ниво. Това ще позволява лесно и удобно пускане на заявки за смяна на работни дни към колегите, споделящи нивата им ако служител няма възможност да се яви. Системата позволява изчисление на оптималния работнически график на всички работници спрямо нуждата от персонал в даден ден, определяна от администратора. При липсата на персонал в даден момент, алгоритъма попълва автоматично с работници, на база на техните изработени часове до момента и/или редакциите на администратора.
* **Изчисление на заплатата –** Модул, позволяващ пресмятането на месечната работна заплата според дните отсъствие/присъствие, данъците и допълнителните доходи или комисионни, определени от работата и приноса на служителя към компанията, които се записват в работническия му профил (ако компанията дава възможност за такъв тип приходи). Страницата е достъпна и в администраторски вариант за допълнителни промени или поправки за всеки един работник.
* **Профил –** където са отразени данните на служителя, неговото ниво и нужна административна информация. Тук имаме записани и нужни контакти и връзки, които той може да използва при нужда.
* **Документи –** Модул за търсене и работа с фирмени документи. Според нивото си, работникът ще има достъп до определени файлове от базата данни, които биха му били необходими. Всеки разполага с документите, означени с неговото, или по ниско ниво. С тази система се цели универсално решение за работнически файлове, но и се запазва конфиденциалността при по-важни документи.
* **Форум –** Прост интерфейс за споделяне на въпроси и теми за разговор в работната среда. В него, хората ще могат да споделят своите питания, както и да разучават вече решени проблеми, с което тяхната работа да се улеснява.

Всички допълнителни модули ще бъдат лесно интерируеми чрез модели и интерфейси, а работния екип на фирмата-клиент ще бъде доволен и информиран, относно всички процеси, засягащи тяхната работа.

За реализацията на проекта ще са нужни 3 отделни сървъра. Единият ще се използва за съхранение на  
информацията в база данни. Вторият ще служи за основната бизнес логика на приложението (Backend). Третия ще се използва за визуализация (Frontend).

 Актьори

* **Администратор**

1. Регистрация на лица със следните данни:
2. Име и фамилия
3. ЕГН
4. Телефонен номер
5. Позиция
6. Заплата
7. IBAN
8. Изготвяне на работен график
9. Управлява потребителските акаунти на всички роли в системата
10. Управление на форума
11. Управление на заплатата на служителите
12. Добавяне на публични и персонални документи
13. Добавяне на контакти

* **Служител**

1. Идентифициране в системата чрез потребителско име и парола.
2. Променя на паролата.
3. Чрез форума задава/отговаря на въпроси свързани с работната дейност, вижда както неговите така и на останалите служители въпроси/отговори
4. Спрямо позицията, която заема ще може да качва файлове (напр. Мениджъра има достъп до тази функционалност, но обикновеният служител/работник, няма), да сваля файлове и да търси файлове.
5. Вижда графика си с работните дни и може да го променя.
6. Вижда всички важни и нужни контакти.
7. При новости/като напр.: отговорен въпрос/, получава известия.

 Използвани термини и символи

* MVC - Model–View–Controller архитектура

 Източници

* Лекции по Софтуерни Архитектури
* <http://www.tu-varna.bg/tu-varnaknt/images/tutorials/ps/lab2.pdf>

Архитектурен обзор

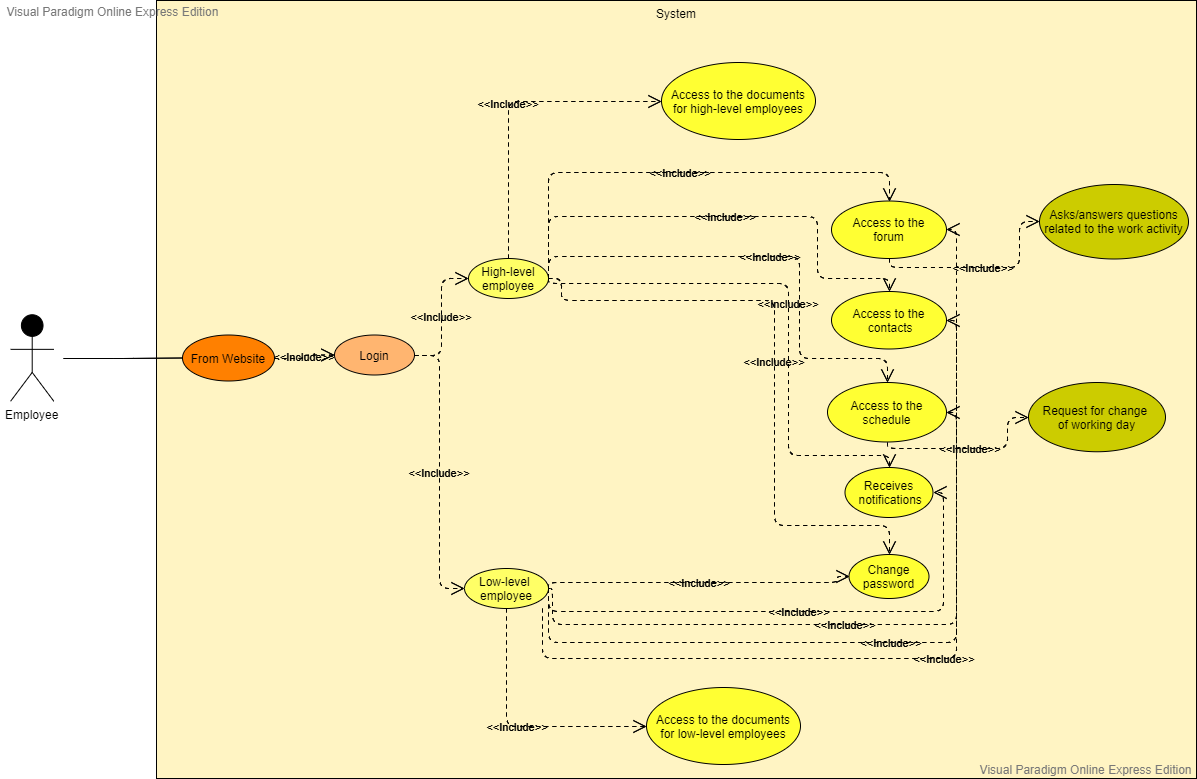
* **Use - case изглед**

 Use cases (Случай на употреба) е техника за определяне на функционалните изисквания на една система.

Сценарият представлява серия от стъпки, описващи взаимодействието между потребител и система. Use case представлява набор от сценарии, свързани помежду си от обща потребителска цел.

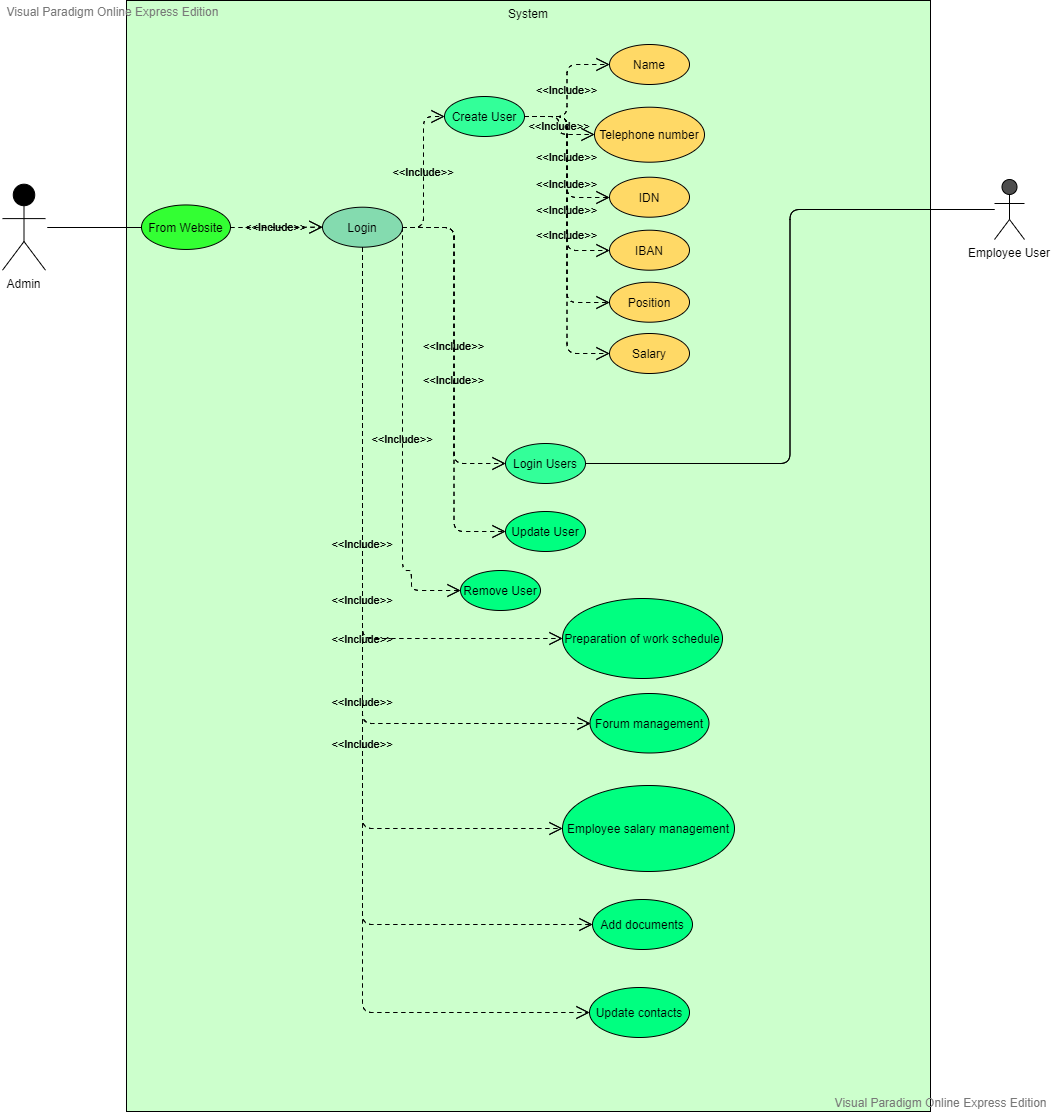
Актьорът (actor) е роля, която един потребител изпълнява по отношение на системата. Всеки use case има основен актьор, който изисква от системата предоставяне на услуга. Основният актьор е актьорът с целта, която use case предписанието трябва да задоволи, и обикновено, но не винаги, е инициатор на use case. Може да има и други актьори, с които системата комуникира в рамките на use case.

Всяка стъпка, наречена още use case предписание, е елемент на взаимодействие между актьор и системата.



***Фигура 1***

На ***фигура 1*** е представена ролята на Employee. Понеже приложението е Website, той се логва чрез него. Спрямо това каква длъжност заема Employee, се определя като служител от високо или ниско ниво. Поради този факт и достъпът до документи е разделен като документи за високо ниво служители и документи за ниско ниво служители. Независимо от нивото на длъжността всеки един Employee има достъп до форума, където може да осъществява комуникация с останалите служители, както и достъп до работният график, където може да прави заявки за промяна на работния ден. Освен това достъп до контакти, промяна на парола и получаване на известия.

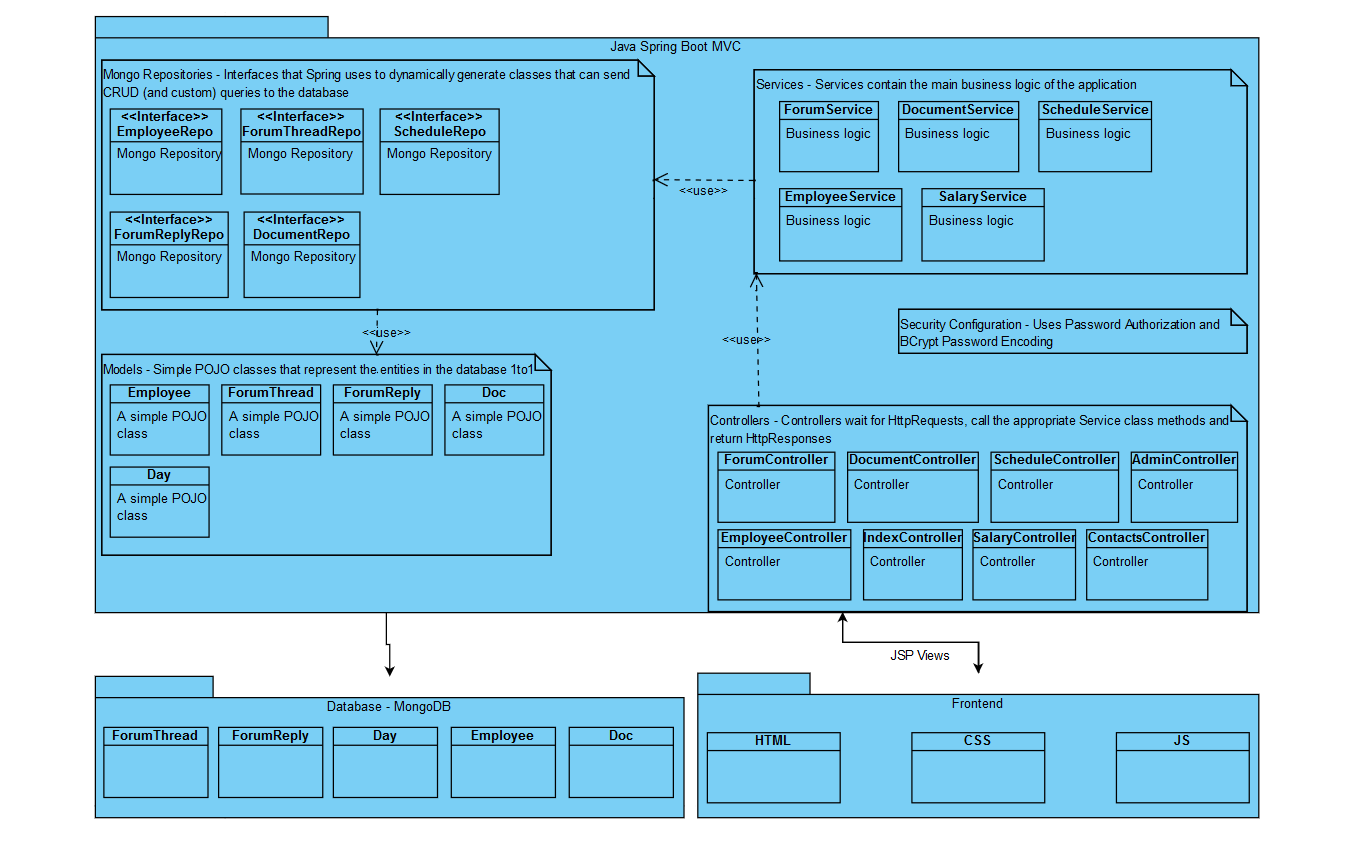


***Фигура 2***

На ***фигура 2*** е представена ролята на Admin. Той се логва, чрез Website. Създава акаунти на служителите съответно с техните данни- Име, Тел.номер, ЕГН, IBAN, Длъжност, Заплата. Освен това може да update-ва и изтрива акаунт на служител. Изготвя работният график на служителите, управлява форма и заплатата на служителите, като прави корекции, ако са нужни, update-ва публичните и персонални документи и контактите.

* **Логически изглед**

Логическият изглед има за цел да представи софтуера като композиция от модули класове. За целта се използват class диаграми които илюстрират връзките между архитектурно зависимите компоненти.



*Приложението е разделено на 3 основни компонента:*

- **Database** – реализирана чрез MongoDB

- **Backend** – реализиран чрез Java Spring Boot

- **Frontend** – реализиран чрез HTML/CSS/JavaScript

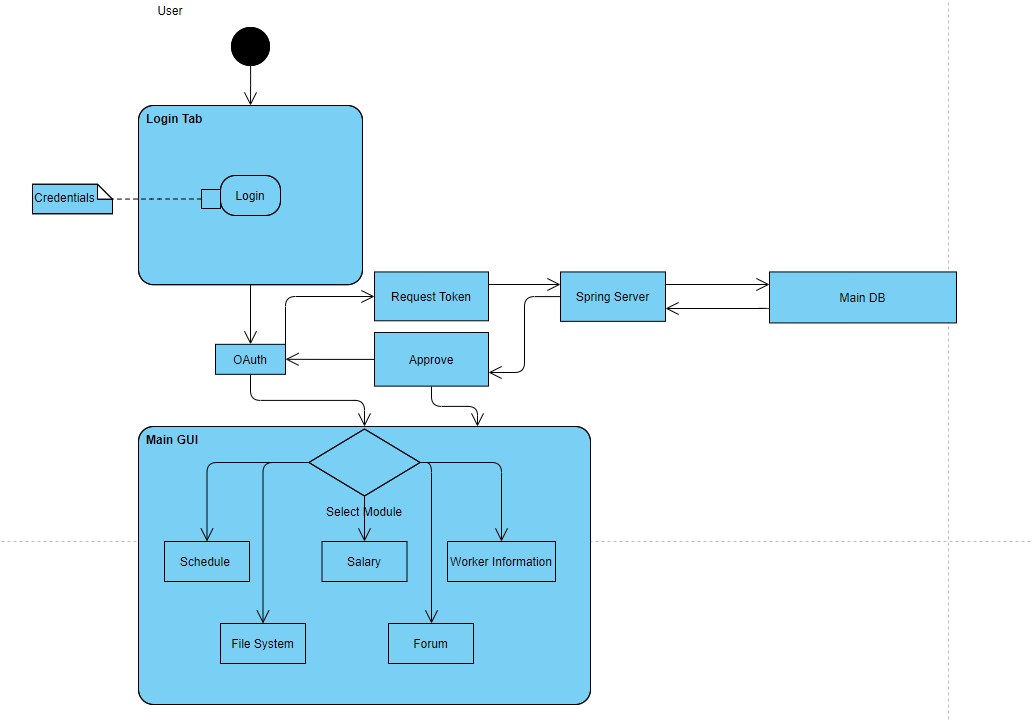
***Backend-a*** *е разделен на 4 основни части:*

- **Models:** Представляват 1 към 1 презентация на таблиците в базата, с помощта на които Spring Boot ще я създаде чрез Code First Approach.

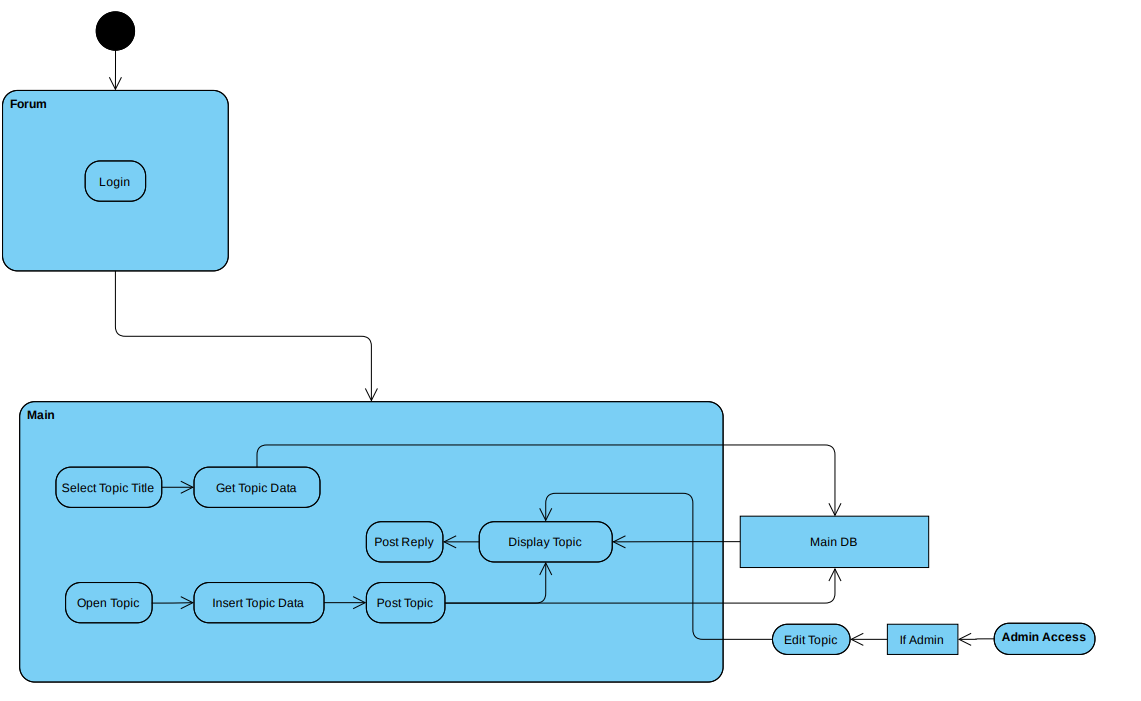
- **Mongo Repositories:** Предоставят възможност за лесно генериране и изпращане на заявки към базата.

- **Services:** В тях се съдържа основната бизнес логика на приложението.

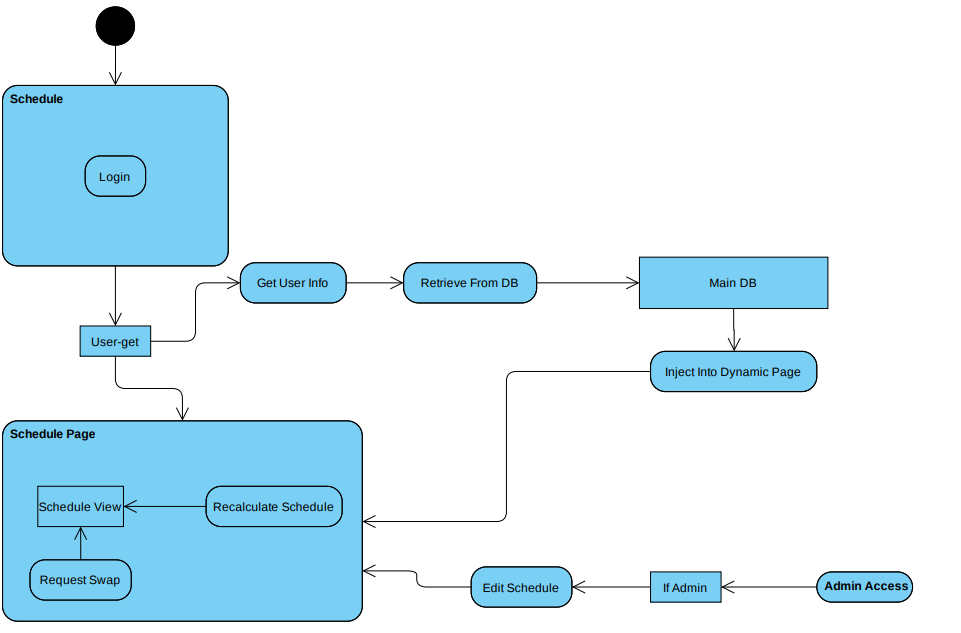
- **Controllers:** Частта от приложението което се занимава с приемането, обработването на заявки от frontend-a, обработването и връщането на view-та на потребителя.

* **Процесен изглед**
* **Главна схема при успешен работнически логин:**

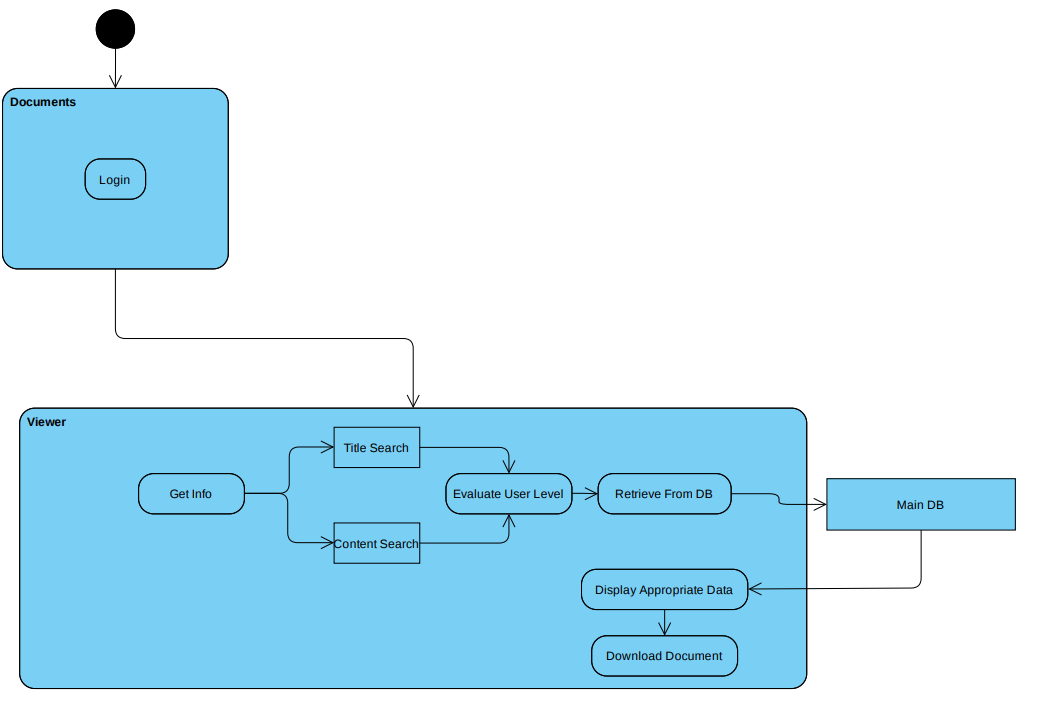
След заявка към сървъра чрез OAuth протокола, служителя получава достъп до своята главна страница на системата. Профилите в нея са служебни и се дават готови на новопостъпилите работници. Сървърът зарежда необходимата информация в страницата и тя се предоставя на потребителя.

* **Изглед на модул „Форум“**

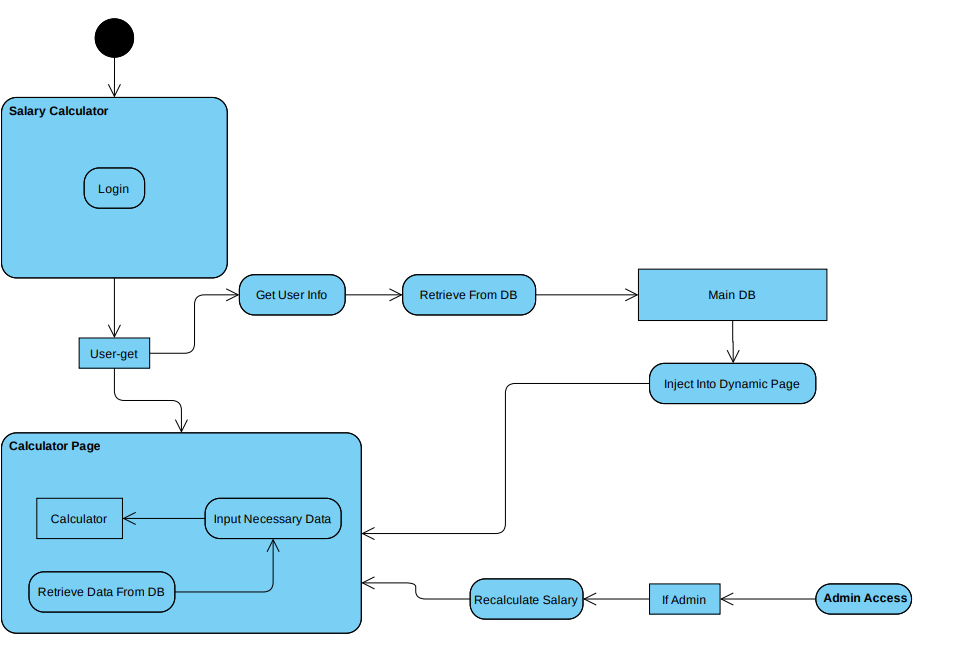
Потребителят има опциите да намира или отваря теми във системния форум. Темите са достъпни според нивата на потребителите и са маркирани цветово според легендата, свързваща цвят с ниво на достъп. Администраторския достъп позволява промяна или премахване на въпросите във форума, както и достъп до всички такива, с цел регулация и централизирано управление на модула.

* **Изглед на модул „График“**

Глобалния график на всички служители се съдържа в базата данни. Според нивото на конкретния служител, той има достъп до всички на неговото, или по ниско ниво. След ограничената заявката до базата данни, съврърът зарежда необходимата информация в страницата. Презареждането на графика става автоматично след всяка промяна. Потребителя има възможност да наблюдава графика, както и да отправя заявка до колегите си за смята на работните им дни. Администратора има универсален достъп и може да променя и преконфигурира графика според нуждите на компанията.

* **Изглед на модул „Документи“**

Тук служителя има възможността да търси и сваля необходимите му документи. След определяне на типа заявка, при него се зареждат файловете му, отново според служебното му ниво на достъп. Опциите му за тях са изглед или сваляне на конкретния документ. Съществува опцията за read-only файлове ако компанията се нуждае от такъв тип достъп. Те могат да бъдат разглеждани, но не и свалени с цел конфиденциалност и защита на фирмата.

* **Изглед на модул „Заплата“**

Предоставя възможност за изчисление на работническото възнаграждение според неговата договорена заплата и/или допълнителните му заслужени доходи. След комбинация на данни от базата и лично въведени такива, потребителят може да разбере ръста в доходите си ако той изпълни дадена избирателна задача - как това ще повлияе на заплатата му. Чрез това, той може да прецени дали има възможност и желание да се захване с нея. Администраторския достъп в случая управлява базата данни и допълнителните постижения на даден работник. Той може да променя възнаграждението за тях или цялостно да премахва/добавя изпълнени задачи.

**Изглед на данни**

Представя модел на обработваните данни, включващ логическите връзки между тях. Ако приложението използва база от данни, се представя схемата на базата от данни, включваща типовете данни, релации и др.

Софтуерът ще използва голяма база данни с различни функционалности. Данните, които са важни за обектите ползващи софтуерът ще бъдат:

На първо място таблица за потребителите:

Users

* Id: text
* First Name: text
* Family Name: text
* Email: text
* Mobile Number: text
* Password: text
* AccessLevel: text
* Salary: double
* EGN: text
* IBAN: text

Тук е важно да отбележим, че таблицата на потребителите има един важен атрибут и това е “Role”, което определя позицията на дадения user (администратор, мениджър, служител и др.). Също така таблицата на потребителите имат атрибутите “Email” и “Password”, с които потребителите биха могли да влизат в своите акаунти. Тази таблица е свързана с таблиците за “Role”, „Graphic“, „Document“, „Forum“…

Следващата таблица е:

Roles

* Id: text
* Name: text

В тази таблица се въвеждат съответните роли, които администрацията предлага. Тази таблица е свързана с “Users”.

Следващата щата таблица е:

Schedule

* Id: text
* Day: day
* EmployeeReferences: text

Тази таблица се отнася за графика за работа за съответния ден. Таблицата съдържа няколко по-важни атрибути като “Employees\_Id”, което е Id-тата на работниците, които трябва да дойдат на работа и “Tasks\_to\_do”, което е Id-тата на задачите, които трябва да се изпълнят за днес. Тази таблица е свързана с таблиците за „Users” и “Tasks”…

Следващата таблица е:

Tasks

* Id: text
* TaskTitle: text
* TaskBody: text
* TaskReceiverID: text

Тази таблица съдържа създадени задачи по дати, които се използват в “Graphic”. Тази таблица е свързана с таблицата за “Graphic”.

Следващата таблица е:

Document

* Id: text
* AccessLevel: text
* UploaderID: text
* DownloaderIDs: List<text>
* Path: text
* Date: String

Тази таблица съдържа атрибути Access за право на достъп кои роли имат право да достъпват този документ. “From\_id” и “To\_id” са атрибути, които съдържат id-тата на потребителя, който е качил този документ и id-то на потребителя, до който трябва да стигне този документ. Атрибутът “Document” е пътят където документът се намира. Таблицата е свързана с “Users” и “Roles”

Следващата таблица е:

Forum

* Id: text
* From\_id: int
* Question: text
* Answer: text
* Answer: text
* Answer: text

…

Тъй като базата данни ще бъде нерелационна таблицата би могла да има различен брой атрибути на всеки ред. Това е таблица за Forum, където даден служител задава въпрос и служителите биха могли да му отговорят. Таблицата е свързана с “Users”.

Нефункционални изисквания

* **Сигурност** - Сигурността се гарантира, чрез използването на SSL/TLS сертификати за енкрипция на данните, разменяни между клиентите и сървъра, както и от OpenAuthorisation протокол за проверка на потребиталския вход.
* **Разширяемост** - Чрез използването на комбинация от MVC и микропроцесна архитектура, системата гарантира лесна и удобна вертикална разширяемост. Проектирането на интегрируеми модели на подсистемите, както и експлицитното дефиниране на кода позволява сигурност при добавяне на функционалност.
* **Достъпност** - Вътрешното кеширане на потребителски сесии позволява възвръщане на етапа, до който той е стигнал. При невъзможност за обработка на Exception и наложителен рестарт на софтуера, сесиите се презареждат до възможно най-късния етап с цел удобство на потребителя.
* **Използваемост** - Интуитивният графически интерфейс улеснява потребителската работа, поради възможността си да представя операциите и действията, извършвани от него. Чрез визуализацията на процесите, които се извършват, потребителя има усет за нужното време за работа на системата и е информиран, относно нейните действия.
* **Производителност** - Системата ще оползотворява ресурсите си, чрез разпределяне на товара с възможност за multithreading.
* **Възможност за тестване** - Системата позволява тестване на работния поток, максималното натоварване, скоростта на обработка на данните и ease-of-access върху отделните аспекти от софтуера. Провежданите тестове ще проверяват качеството на backend логиката, както и на UX дизайна.