



## Atlas Service Management



Мониторинг на разпределените задачи в експеримента Атлас в ЦЕРН

ЦЕРН - НАУКА ЗА МИР!



1. **ТЕМА:** Atlas Service Management

2. **АВТОРИ:**

Алекс Василев Милчов, ЕГН: 0246043829, Телефон: 0893762009; гр. Перник, ул. Отец Паисий №61; ПГ по икономика – гр. Перник, XII клас

Давид Димитров Иванов, ЕГН: 0250313781, Телефон: 0896239867; гр. Перник, ул. Струма №19; ПГ по икономика – гр. Перник, XII клас

3. **РЪКОВОДИТЕЛ:** Людмил Велинов Любомиров – Старши учител по теоретично обучение в ПГ по икономика – гр. Перник, тел.: 0895481106, e-mail: [lusivelinov@abv.bg](mailto:lusivelinov@abv.bg)

4. **РЕЗЮМЕ:** Проектът представлява мониторинг на системата за разпределени изчисления на експеримента ATLAS в ЦЕРН и показва текущото състояние и натоварване на компютърните ресурси, колко процесорни ядра работят в момента в целия свят и др. Целият процес е визуализиран чрез интерактивни диаграми и таблици.

4.1 **Цели** Целта на приложението е да се наблюдава разпространението на изчисленията на експерименталния ATLAS в ЦЕРН и да се прегледа текущото състояние и натоварване на компютърните ресурси, колко CPU са налични в момента и др.

4.2 **Ниво на сложност:** Проектът е разработван в продължение на 7 месеца.

**Екип:**

1. Алекс Василев Милчов – разработка, тестване, конфигуриране и управление на уеб приложението.

2. Давид Димитров Иванов – разработка, тестване, конфигуриране и управление на сървърната част от проекта.

4.3. **Основни етапи в реализирането на проекта:**

1. Установяване на темата на разработка

2. Уточняване на задание с представител от ЦЕРН

3. Събиране на необходимите ни данни и материали
4. Проучване на данните от Atlas Experiment и BigPanDA
5. Изграждане на функционален план
6. Избор на технологии
7. Изграждане на логически план
8. Разработка, тест и публикация

#### 4.4 Езици и технологии:

##### - езици:

**PHP** - PHP е един от най-широко използваните програмни езици за създаване и реализиране на уеб услуги. Поради отличната поддръжка на редица бази данни като MySQL, Oracle и PostgreSQL, PHP много често се използва съвместно с бази данни. В добавка, вградените над 800 функции и наличието на отделни библиотеки с функции правят PHP език с широки възможности.

**JavaScript** – JavaScript, един от най-използваните скриптов езици, даващ възможността за динамична промяна на съдържанието на страницата. Поддържа обектно-ориентиран и функционален стил на програмиране. Чрез популярността си през годините, се изграждат най-различни библиотеки и правят езика доста по-иновативен.

**JSON** - JSON (JavaScript Object Notation) е опростен формат за обмяна на данни, удобен за работа както за хората, така и за компютрите. Той е базиран на едно подмножество на езика за програмиране JavaScript. JSON има текстов формат, напълно независим от реализацията на езика, но използва конвенции, които са познати на програмистите на C-подобни езици, включително C, C++, C#, Java, JavaScript, Perl, Python, и много други. Тези свойства правят JSON идеален език за обмяна на данни.

**SQL** - SQL е структуриран език за заявки. Аббревиатура от Structured Query Language. С негова помощ можем лесно да правим запитвания и заявки към релационни бази данни. Това е език с две основни функции - с него можем да дефинираме структури от данни и също така можем да обработваме данните, които искаме да запишем в базата данни и/или вече са записани.

##### - технологии:

**Composer** - Composer е мениджър на пакети на ниво приложение на езика за програмиране PHP, който предоставя стандартен формат за управление на зависимости в PHP софтуер, както и необходимите библиотеки.

**Laravel** - Laravel е PHP фреймуърк (framework) с отворен код, базиран на шаблона MVC (model–view–controller), който дава възможност за разделяне на бизнес логиката от графичния интерфейс. Архитектурата разделя изходния код на три отделни части – модел, изглед и контролер, което позволява лесна модификация и структуриране на кода.

**Eloquent** – Eloquent е ORM (Object Relational Mapper), който осигурява един лесен начин за комуникация с базата данни. При Eloquent, всяка таблица на базата данни има съответния „Модел“, който се използва за взаимодействие с тази таблица.

**MySQL** - MySQL е релационна база данни. Използва SQL (Structured Query Language) - най-популярния език за добавяне, достъпване и обработване на информация в база от данни. Базата е с отворен код (open source), може да бъде свалена от Уеб и използвана от всеки според неговите нужди, без да има нужда от лицензиране. Предимството на MySQL е, че специфичната информация може да бъде съхранявана в отделни, много на брой малки таблици с установени между тях се връзки (релации), вместо в една единствена огромна таблица. С това много се улеснява и ускорява обработката на информацията.

**VueJS** - VueJS е JavaScript фреймуърк (framework), използван за изграждане на уеб приложения, с отворен код, базиран на шаблона MVVM (model–view–viewmodel), който дава възможност за разделяне на бизнес логиката от графичния интерфейс. Чрез архитектурата, изходният код се разделя на три отделни части – модел, изглед и изглед-модел, което позволява лесна модификация и структуриране на кода.

**Vuex** - Vuex е State Management Pattern, който служи за централизация на всички компоненти в приложението, с различни правила, които гарантират, че състоянието му може да бъде мутирано само по предсказуем начин.

**VueFormulate** – VueFormulate е библиотека, която предлага удобен начин за изграждане на форми с Vue. Създадена е за улеснение на разработчиците, с настройки, които правят внедряването на общи функции изключително лесно.

**Chart.js** – Chart.js е JavaScript библиотека, с отворен код, която предоставя едни различни, гъвкави и интерактивни, диаграми за потребителя.

**Tailwind CSS** – Tailwind CSS е CSS фреймуърк, който предоставя задълбочен списък на CSS класове и инструменти, които позволяват лесно изграждане на оформлението на проекта.

**Space** – Space има много функции, предназначени за процеса на разработване на софтуер. Има инструменти за поддържане на CI / CD, преглед на код Система за контрол на версиите (Git), автоматизация на изграждането и внедряването на приложения и регистри на пакети и контейнери за публикуване на екипни дейности.

**GitHub** – GitHub е уеб базирана услуга за разполагане на софтуерни проекти и техни съвместни разработки върху отдалечен интернет сървър в т.нар. хранилище (software repository).

**Sourcetree** – Sourcetree ни дава възможност за едно доста по-лесно взаимодействие между различните Git хранилища, а също така ни позволява и удобен начин за визуализация и управление на хранилищата чрез Git GUI на Sourcetree.

## 4.4. Главни проблеми при разработка:

### - сървър:

1. Подредба на код – Подреждането на кода и спазването на конвенциите за правилно написан код не беше толкова просто задача, защото добрите практики са много. Трябваше да измислим начин, с който да постигнем ефект на гъвкавост, за да можем в бъдеще да разширим много по-лесно сървърната част. Решихме този проблем с помощта на най-различни шаблони за дизайн (design patterns).

2. Структура на базата данни – Структурирането на базата данни също не беше лесно нещо за нас, защото трябваше да помислим за добър начин, по който можем да направим таблиците в базата данни. Искахме да я направим възможно най-проста, за да може да работим много по-лесно с нея. Успяхме да го направим чрез моделите в Laravel, с които установихме, че ще ни бъде много по-лесно.

### - уеб приложение:

1. Подредба на код – Подреждането на кода и спазването на конвенциите за правилно написан код не беше толкова просто задача, защото добрите практики са много. Трябваше да измислим начин, с който да постигнем ефект на гъвкавост, за да можем в бъдеще да разширим

много по-лесно потребителската част от уеб приложението. Уеб приложението се поддържа от функционалност за „контрол на достъпа“, затова трябваше да напишем качествен и автоматизиран код. Решихме този проблем с помощта на най-различни шаблони за дизайн (design patterns).

## 2. Дизайн на потребителския интерфейс (UI)

Главната ни цел при разработката на качествен UI бе да достави максимално удобство на потребителя, без това да се отразява негативно на неговата ефективност. Ето кои са стъпките, през които протича процесът при създаването на UI дизайна на приложение:

- **Събиране на изискванията на функционалността** - създадохме списък с функционалностите, които ще поддържа нашата системата, за да се постигнат максимално целите на проекта и потенциалните нужди на потребителя.

- **Анализ на потребителите и задачите** – на тази стъпка трябваше да проведем проучване, което включва анализ на потенциалните потребители на системата, като трябваше да изучим начина, по който те извършват задачите, които функционалностите на дизайна предоставя.

- **Информационна архитектура** - разработката на процеса и/или информационния Проект №86 – Atlas Service Management поток на системата (Three flowchart), който показва йерархията на отделните страници и екрани, от които е създадено уеб приложението.

- **Прототип** - разработихме кадри, които показват визуално елементи, от които ще се състои интерфейса. На лист хартия начертахме основен дизайн, както и отделни компоненти, от които ще се състои нашия потребителски интерфейс.

- **Тестване на използваемостта** - изпробване на прототипи върху обикновен потребител – често се използва техника, наречена „think aloud protocol“, където вие карате потребителя да сподели своите впечатления по време на тестването. По този начин UI тестването позволява на дизайнера да разбере дизайна си, по начина, по който го разбира потребителят и така по-лесно да създаде успешни приложения.

- **Графичен UI дизайн** – тук вече изготвихме актуалния изглед към финалния графичен потребителски дизайн (Graphic user interface – GUI). Той се базира на данните, придобити по време на етап “Анализ на потребителите и задачите”, но претърпя известни промени, наложени от резултатите от “Тестването на използваемостта”. В зависимост от типа на интерфейса, който

създавахме, процесът включваше и програмиране, чрез което накарахме системата да изпълни дадено действие.

- **Софтуерна поддръжка** – След пускането на новия интерфейс, регулярна поддръжка може да бъде необходима, за да се отстранят софтуерни бъгове, да се променят опции или да се направи цялостен ъпгрейд на системата. Ако обаче се стигне до ъпгрейд на интерфейса, той задължително трябва да премине наново през всички гореописани стъпки.

Потребителският интерфейс е графичното оформление. Състои се от бутоните, оформлението на текста, изображенията, плъзгачите, цветовете, шрифтовете и всички останали елементи, с които потребителят взаимодейства в приложението. UI дизайнът включва оформление на екрана, преходи, анимация на интерфейса и всяко отделно микро-взаимодействие. Тази работа се оказва изключително сложна, тъй като трябваше да доставим максимално добро изживяване на обикновения потребител. Проблемът беше решен като избрахме една от най-актуалните технологии за изработката на динамичен дизайн на потребителския интерфейс – [Tailwind CSS](#). Този фреймуърк съхранява готови компоненти и класове за изграждане на дизайн. Технологията позволява написването на качествен код като позволява директното написване в основната структура на дадената уеб страница.

```
<div class="lg:pt-12 pt-6 w-full md:w-4/12 px-4 text-center">
  <div
    class="relative flex flex-col min-w-0 break-words bg-white w-full mb-8 shadow-lg rounded-lg"
  >
    <div class="px-4 py-5 flex-auto">
      <div
        class="text-white p-3 text-center inline-flex items-center justify-center w-12 h-12 mb-5 shadow-lg rounded-full bg-red-400"
      >
        <i class="fas fa-award"></i>
      </div>
      <h6 class="text-xl font-semibold">"What is ADC?"</h6>
      <p class="mt-2 mb-4 text-gray-600">
        ADC is ATLAS Distributed Computing - the distributed
        computation system of the ATLAS experiment.
      </p>
    </div>
  </div>
</div>
```

3. Навигационна система – Голяма роля в изработката на уеб приложение е начина на навигация. Всяко приложение само по себе си е една презентация на услуги, продукти или информация. Ако потребителите изпитват трудност да намерят някаква препратка към желаната

от тях информация, то това е лошо потребителско изживяване. Това разбира се не означава половината екран да бъде в менюта или да се прекали с така наречените „залепени менюта“, които да закриват важната за потребителя информация. Това е част от така нареченото понятие “Потребителско изживяване (User experience - UX)”. Проектът “Atlas Service Management” се сблъска с оформянето, конфигурирането, създаването и поддръжката на една навигационна система. Създадохме така нареченият “Sidebar”. Това е вертикален навигационен компонент, който освен традиционните текстови връзки може да вгражда икони, падащи менюта, аватари или формуляри за търсене. По силата на своята яснота и простота този компонент значително увеличава Потребителското изживяване (User experience - UX). Компонентът ви позволява да навигирате бързо както в малки приложения, така и в огромни портали. Неговата функционалност за вграждане на множество връзки ви позволява да внедрите по-усъвършенствана категоризация на съдържанието, което важно при по-големите проекти, какъвто е уеб приложението “Atlas Service Management”.

#### 4.5 Логическо и функционално описание на решението:

- **сървър**: Като за начало инсталирахме Laravel на нашата машина чрез Composer. Файловата структура - стандартна за фреймуърк (framework) като Laravel. За по-качествен код в допълнително създадените файлове от нас, използвахме HMVC (Hierarchical Model-View-Controller) архитектура. Тук са описани по отделно допълнително създадените папки и файлове:

- **app** – централната точка, която съдържа основния код на сървъра въпреки това, почти всички класове се намират в тази директория

- **bootstrap** – директория за зареждане на файла app.php, който зарежда Laravel. Тази директория съдържа и кеш директория, която съдържа файлове, генерирани от Laravel за оптимизация на производителността, като файловете за кеш (cache), маршрута (route) и услугите (services)

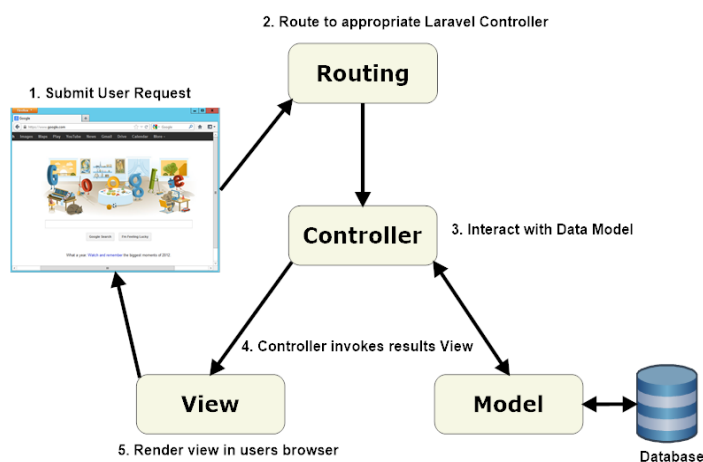
- **config** – конфигурационната директория, както подсказва името, съдържа всички конфигурационни файлове за нашето приложение

- **database** – директорията за базата данни

- **public** – публичната директория съдържа файла index.php, който е входната точка за всички заявки, влизащи в приложението ни



- **resources** – директория с ресурси и езикови файлове
- **routes** – директория на маршрутите (routes). Най-важният файл в тази директория е api.php, който съдържа всички мрежови адреси на дадения ресурс за Application Programming Interface (API)
- **storage** – директория за съхранение на компилираните шаблони на Blade темплейтната система
- **tests** – тестова директория за автоматизирани тестове (Unit Tests)
- **vendor** – директория, която съдържа composer зависимости



## - уеб приложение:

За инсталацията на Vue проект, използвахме помощния инструмент за библиотеки на Javascript – NPM. С негова помощ бързо създадохме проект и започнахме работа. За качествен код в допълнително създадените файлове от нас, използвахме MVVM (View-ViewModel-Model) архитектура. Тук са описани по отделно допълнително създадените папки и файлове:

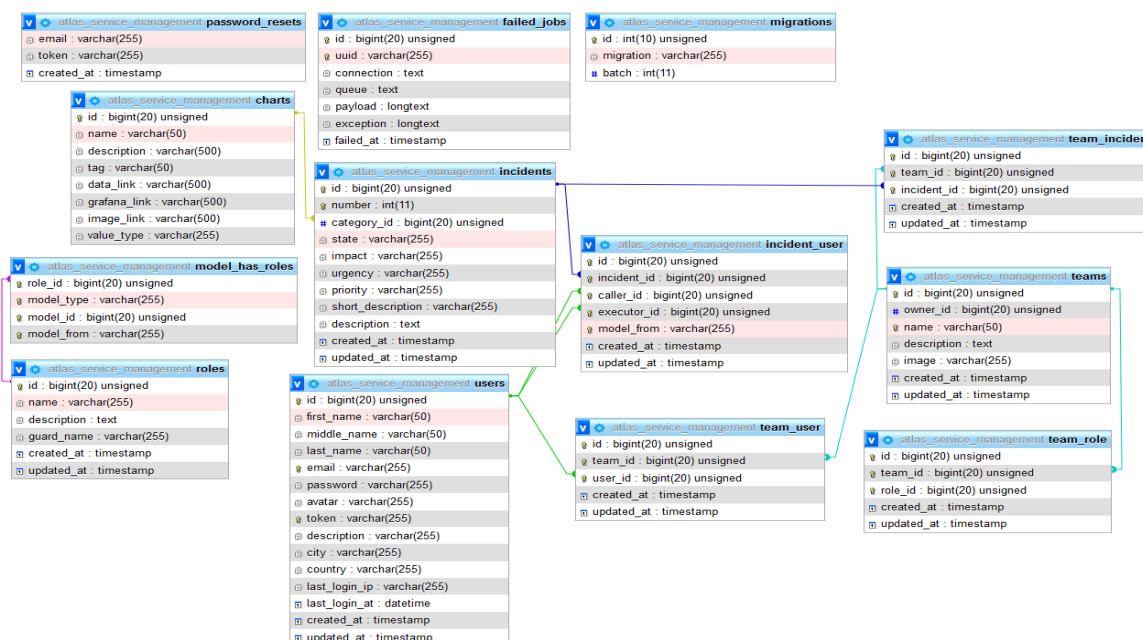
- **node\_modules** – папка, съдържаща модули, които биват използвани
- в главната директория се намират файлове, използвани за дефинирането и наблюдението на зависимости и библиотеки (composer.json, composer.lock, yarn.lock, package.json, package-lock.json, vue.config.js, tailwind.config.js, babel.config.js)
- **src** – главна директория за всички файлове на VueJS, контролиращи уеб приложението, както и всички налични компоненти и библиотеки.
- **api** – персонализирана директория за библиотеката axios. Съдържа всички файлове, готови за изпълнението на HTTP заявки.



- **assets** – директория за мултимедийно съдържание. Съдържа файлове като снимки, стилове за дизайн на приложението и различни файлове с данни.
- **components** – директория, съдържаща всички компоненти на приложението. Компонентите са една от най-мощните функции на Vue.js. Те ви помагат да разширите основните HTML елементи, за да капсулирате код за многократна употреба.
- **layouts** – директория, съдържаща различни слоеве и страници на уеб приложението. Важна част за доброто представяне на потребителския дизайн.
- **models** – класове, използвани при комуникацията между различните слоеве на компонентите.
- **services** – цялата логическа част на приложението. Тук се съдържат всякакви HTTP заявки, бизнес логика на компонентите и заявки към файловете на Vuex. За глобалните променливи.
- **store** – логиката, която съдържа глобалните променливи. Тази папка поддържа основните функционалности на проекта.
- **views** – В тази папка може да не се намери основният потребителски интерфейс. Отправна точка към бизнес логиката на уеб приложението.

## База Данни (БД):

Базата данни се състои от 13 таблици (users, roles, model\_has\_roles, charts, incidents, incident\_user, teams, team\_user, team\_role, team\_incidents, password\_resets, failed\_jobs, migrations):



- **users** – Таблица, която отговаря за потребителите. В нея се съдържат регистрираните потребители и чрез нея потребителите влизат в профила си. Тя притежава следните полета: (id, first\_name, middle\_name, last\_name, email, password, avatar, token, description, city, country, last\_login\_ip, last\_login\_at, created\_at, updated\_at).

- **roles** – Таблица, която отговаря за ролите. В нея се съдържат абсолютно всички роли в приложението. Тя притежава следните полета: (id, name, description, guard\_name, created\_at, updated\_at).

- **model\_has\_roles** – Таблица, която отговаря за ролите на съответния Laravel модел. Базирана на Polymorphic Relationships. В нея се съдържат всички роли, които са добавени и свързани със съответната връзка на самия Laravel модел. Тя притежава следните полета: (role\_id, model\_type, model\_id, model\_from).

- **charts** – Таблица, която отговаря за диаграмите. В нея се съдържа, цялата информация относно диаграмите. Тя притежава следните полета: (id, name, description, tag, data\_link, grafana\_link, image\_link, value\_type).

- **incidents** – Таблица, която отговаря за инцидентите. В нея се съдържа, цялата информация относно инцидентите. Тя притежава следните полета: (id, number, category\_id, state, impact, urgency, priority, short\_description, description, created\_at, updated\_at).

- **incident\_user** – Таблица, която отговаря за инцидентите на потребителите. В нея се съдържат всички инциденти, които са създадени и изпълнени от съответните потребители. Тя притежава следните полета: (id, incident\_id, caller\_id, executor\_id, model\_from, created\_at, updated\_at).

- **teams** – Таблица, която отговаря за отборите. В нея се съдържа, цялата информация относно отборите. Тя притежава следните полета: (id, owner\_id, name, description, image, created\_at, updated\_at).

- **team\_user** – Таблица, която отговаря за членовете в отборите. В нея се съдържат всички членове, които са добавени в съответния отбор. Тя притежава следните полета: (id, team\_id, user\_id, created\_at, updated\_at).



- **team\_role** – Таблица, която отговаря за ролите в отборите. В нея се съдържат всички роли, които са добавени в съответния отбор. Тя притежава следните полета: (id, team\_id, role\_id, created\_at, updated\_at).

- **team\_incident** – Таблица, която отговаря за инцидентите в отборите. В нея се съдържат всички инциденти, които са добавени в съответния отбор. Тя притежава следните полета: (id, team\_id, incident\_id, created\_at, updated\_at).

- **password\_resets** – Таблица, която идва по подразбиране с Laravel и отговаря за сменените пароли на потребителите. Тя притежава следните полета: (email, token, created\_at).

- **failed\_jobs** – Таблица, която идва по подразбиране с Laravel и отговаря за задачите (jobs) в опашките (queues), които също са част от Laravel. Тя притежава следните полета: (id, connection, queue, payload, exception, failed\_at).

- **migrations** – Таблица, която идва по подразбиране с Laravel и отговаря за миграциите (таблиците). В нея се съдържат имената на таблиците, които сме създали. Тя притежава следните полета: (id, migration, batch).

## 4.6 Реализация:

При писането на PHP и JavaScript кода е използван редактор Visual Studio Code, в комбинация с различни разширения за оптимизиран работен процес. MySQL базата данни се администрира с PHPMyAdmin. Групирахме различните заявки към нашето RESTful API, написано на фреймуърка Laravel, под различни директории и файлове за по-лесен ориентир и надграждане по време на разработката на платформата.

Системата е изградена с повишено внимание към файловата и логическа структура. Тъй като имаме изключително много идеи за доразвитие на системата, беше нужна солидна и лесна за поддръжка основа, каквато смятаме, че сме изградили. За да работим съвместно използваме система за управление на версиите Git /Version Control/ и място за съхранение на сорс кода GitHub. По време на реализацията много акцентираме върху комуникацията, тъй като всеки от нас се специализира в конкретни области и често се налага взаимно да обменяме опит, както и да вземаме важни решения, чийто последствия да бъдат решаващи за в бъдеще.

В процеса на изпълнението на задачите ние постоянно тествахме, оптимизирахме, променяхме и дори пренаписахме части от платформата, с цел най-максимална производителност и максимално удобство на потребителя.



Разучаването на технологиите играеше основна роля в началото на направата на нашата платформата, тъй като ние вдигнахме нашите персонални летви и от нас се изискваше да бъдем постоянни и с всеки изминал ден да усъвършенстваме нашите възможности и начин на мислене.

Качването на сървъра и уебсайта премина през доста етапи. Поради грешки в PHP кода, поради неправилна версия на PHP, трябваше да „смъкнем“ версията на PHP на проекта, тъй като инсталираната версия на PHP на персоналните ни компютри бе по-нова от тази на хостинга.

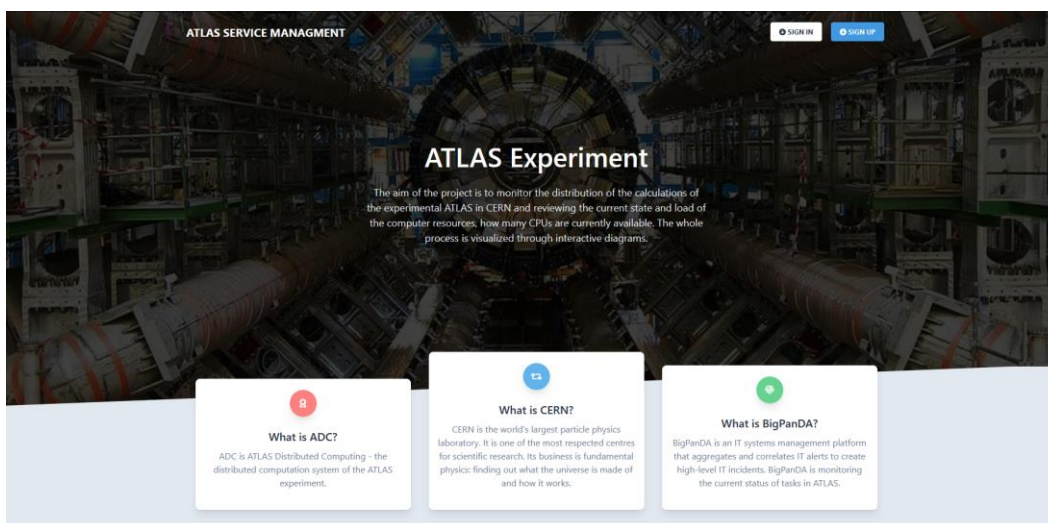
Сайтът работи под браузърите Google Chrome, Mozilla Firefox, Opera, Microsoft Edge, Safari, Brave.

#### 4.7 Описание на приложението:

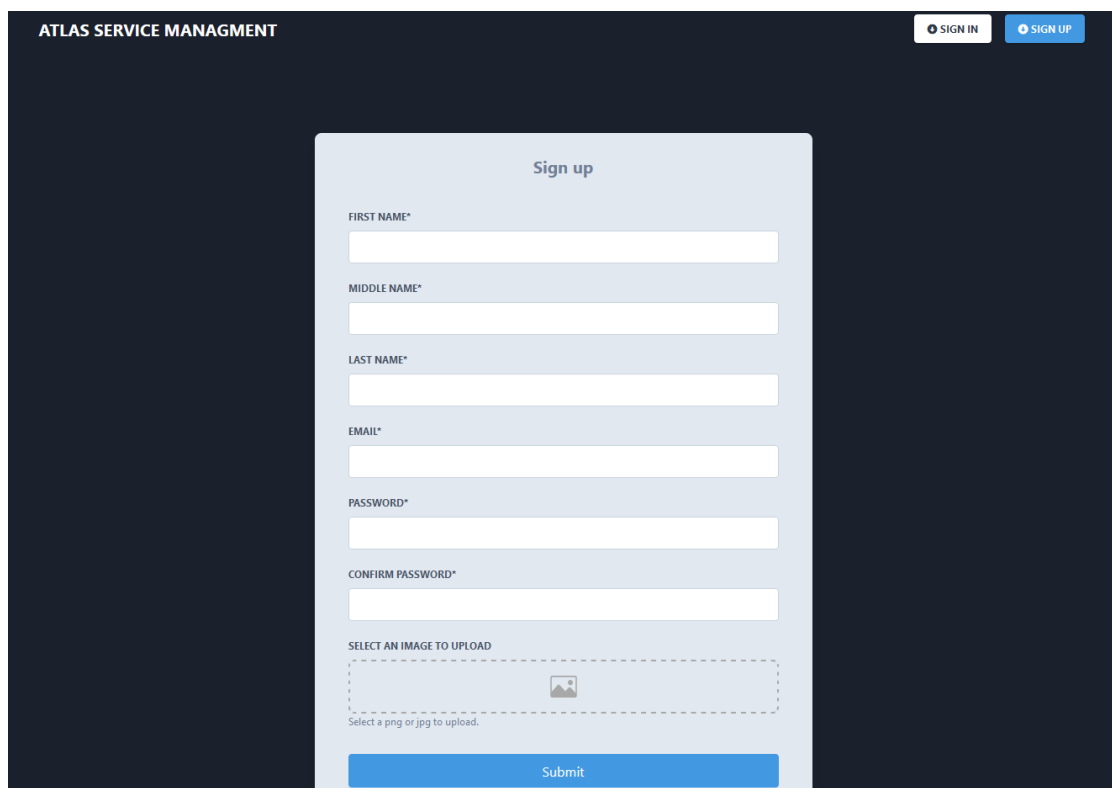
Приложението може да бъде видяно тук: <https://atlas.noit.eu>

„Atlas Service Management” е уеб приложение, което ще бъде използвано за научните цели в ЦЕРН. Неговите цели са съсредоточени върху малка научна аудитория, която познава устройството на експеримента ATLAS.

При зареждане на главната страница най-отгоре можем да видим навигационното меню (navbar-a), на който отляво е поставено наименованието на проекта, а отдясно два бутона „Sign Up” и „Sign In”. Малко по-надолу можем да видим снимка на адрония колайдер в ЦЕРН, с кратко описание за това какво представлява проекта. След това ни се показват три карти „cards”, чрез които можем да прочетем различна информация относно ADC, CERN и BigPanDA. По-надолу можем да прочетем за „суровите данни”, които взимаме от BigPanDA, а също така и информация относно експеримента в ATLAS. В предпоследната част от главната ни страница можете да прегледате информация за нас, а в края на страницата контактна форма, чрез която можете да се свържете с нас, ако сте открили някакъв проблем в системата ни.



При натискане на бутона „Sign Up” се показва формата за регистрация, от която потребителят може да си създаде профил, като просто трябва да попълни няколко полета (Име, Презиме, Фамилия, Емейл, Парола, Потвърждаване на парола), а също така може да сложи снимка, ако желае. След като извърши този процес и натисне бутона „Submit”, той автоматично получава ролята „ess” (Employee Self Service), която е базова за всеки потребител от нашата система.



## Примерни акаунти за достъп:

### Admin Role:

Email: admin@admin.com

Password: asmadminpassword

### Employee Self Service Role:

Email: ess@ess.com

Password: asmesspassword

### Manager Chart Role:

Email: managerchart@managerchart.com

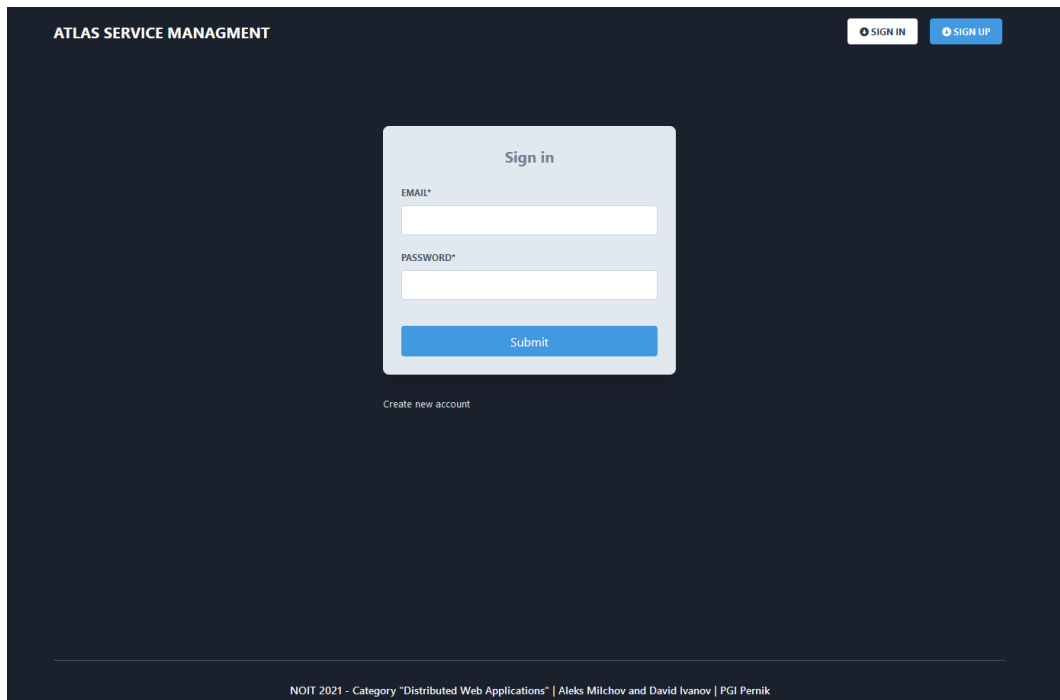
Password: asmmanagerchartpassword

### Manager Table Role:

Email: managertable@managertable.com

Password: asmmanagertablepassword

При натискане на бутона „Sign In” се показва формата за вход, от която потребителят може да влезе в своя профил, чрез (Имейл и Парола). След като влезе в профила си, той бива автоматично препратен на собствения си dashboard.



DASHBOARD/ТАБЛО ЗА УПРАВЛЕНИЕ представлява уеб страница, показваща основните данни на потребителя, с които имаме моментен достъп до системата. Той може да регулира и управлява данните на своя акаунт. Потребителят може да добие представа за своите данни от дясната част на уеб страницата.

- Потребителски аватар или потребителска снимка на профила.
- Три колони на един ред, които съдържат информация за:
  - Броят на ролите на потребителя.
  - Броят на собствените инциденти.
  - Броят на отборите, в които е член.
- Име, Презиме, Фамилия.
- Кога е създал профила си и кога последно го е променял.
- Зададено местоположение:
  - City (Град)
  - Country (Държава)
- Наименованието на всички роли, съдържащи се в дадения акаунт.
- Кратко описание на потребителя.

В централната част на уеб страницата може да се редактират съответните данни:

- Име (FIRST NAME)
- Презиме (MIDDLE NAME)
- Фамилия (LAST NAME)
- Имейл адрес (EMAIL ADDRESS)
- Град (CITY)
- Държава (COUNTRY)
- За мен (ABOUT ME) или Описание за потребителя.



ADMIN ADMIN - DASHBOARD

My account

USER INFORMATION

FIRST NAME

Admin

MIDDLE NAME

Admin

LAST NAME

Admin

EMAIL

admin@admin.com

CONTACT INFORMATION

CITY

Pernik

COUNTRY

Bulgaria

ABOUT ME

DESCRIPTION

New description to my profile.

Update

11

Roles

10

Incidents

22

Teams

Admin Admin Admin

CITY: PERNIK

COUNTRY: BULGARIA

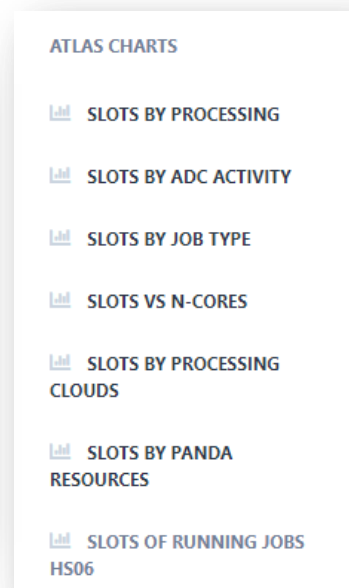
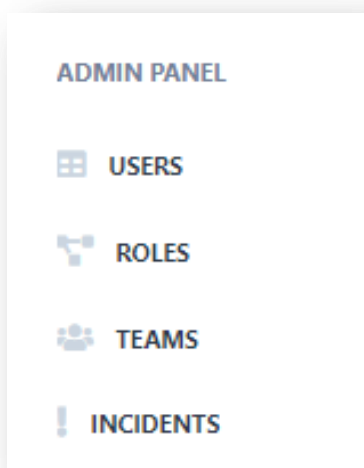
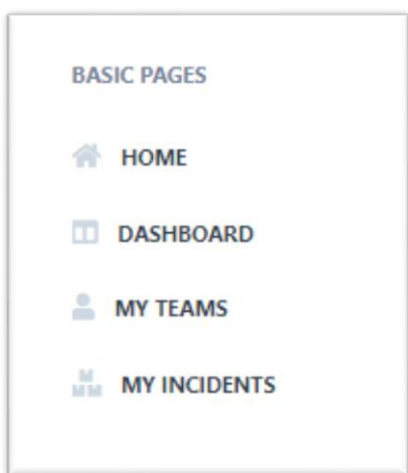
CREATED AT: 2021-03-05 06:36:19

UPDATED AT: 2021-05-03 23:30:29

Roles: manager.chart, manager.chart\_slots.by\_adc\_activity, manager.chart\_slots.by\_job\_type, manager.chart\_slots.by\_processing, manager.chart\_slots.by\_adc\_activity, manager.chart\_slots.by\_job\_type, admin.manager.chart\_slots.by\_panda\_resources, manager.chart\_slots.of\_running\_jobs\_hs06, manager.table\_slots.by\_processing, manager.table\_events\_simul+pile

Description: New description to my

Нашата платформа използва добре познатия начин за навигация, а именно - страничната навигация (sidebar), която според нас е по-удобна и лесна за използване от стандартната. В страничното навигиращо меню на личното табло на потребителя (dashboard) може да бъде избран някой от компоненти в уеб приложение и автоматично потребителя да бъде препратен към него.



Страничното навигиращо меню показва избрани компоненти и навигиращи елементи за съответните страници според ролята на потребителя в системата. Всичките роли, които са добавени и успешно функциониращи в системата са 32 на брой и главните от тях са:

- admin - admin е роля, която се задава на основния потребител. Той има всички разрешения и може да управлява всичко.
- ess - Employee Self Service е роля, която е зададена на потребителя по подразбиране.
- manager\_chart - роля, която се задава на главния мениджър диаграмите. Той може да управлява всичко за диаграмите.
- manager\_table - е роля, която е зададена на генералния мениджър на таблиците.

Страничната навигация е разделена на три подсекции или три категории. Първата от категориите е BASIC PAGES или базови страници. Всяка една от 32-те роли имат достъп до тази подсекция. Категорията съдържа следните връзки към компонентите и уеб страниците:

- Home – Навигиращ хиперлинк, препращащ директно към началната страница на уеб сайта.
- Dashboard – табло за управление на данните на потребителя. Функцията на тази страница описахме по-нагоре в документацията.
- My Teams – Хиперлинк към уеб страница, показващи всички отбори в които членува дадения потребител.
- My Incidents – Препраща към уеб страница, в която има таблица с всички инциденти за потребителя.

Втората подсекция се нарича ADMIN PANEL. Категорията е достъпна само за потребители, които съдържат ролята “admin”. Никой друг потребител няма достъп до тази подсекция. Категорията съдържа следните връзки към компонентите и уеб страниците:

- Users - Погледът в тази страница е насочен директно към таблицата с всичките потребители, регистрирани в уеб приложението. В тази страница потребителят може да сортира спрямо 4 категории – First Name (име), Middle Name (презиме), Last Name (фамилно име), Created (дата и час на създаване на профил), като така може да види кои потребители са създадени най-скоро. Таблицата има и търсачка, с която може да намерите най-бързо търсения от вас потребител според някоя от стойностите в четирите колони. Страницата позволява и

изтеглянето на CSV файл чрез бутон, за да може работата на администратора да бъде максимално улеснена при споделянето или съхраняването на информацията на външен преносител на данни.

OVERVIEW

Users - Data Table

Q

Search User

DOWNLOAD CSV

	First Name	Middle Name	Last Name	Created	Updated
1	Admin	Admin	Admin	2021-03-05 06:36:19	2021-05-03 23:30:29
2	Employee	Self	Services	2021-03-21 12:41:33	2021-05-03 14:03:18
3	Manager	Chart	Chart	2021-03-21 12:41:33	2021-05-03 09:26:28
4	Manager	Tables	Table	2021-03-21 12:41:33	2021-05-03 13:53:43

Rows per page: 10

1 - 5 of 5

Previous


Next

1	id	first_name	middle_name	last_name	email	avatar	created_at	updated_at
2		1 Admin	Admin	Admin	admin@admin.com	https://atlas.noit.eu/storage/av	3/5/2021 6:36	3/5/2021 6:36
3		2 Employee	Self	Service	ess@ess.com	https://atlas.noit.eu/storage/av	3/21/2021 12:41	3/21/2021 12:41
4		3 Manager	Chart	Chart	managerchart@manag	https://atlas.noit.eu/storage/av	3/21/2021 12:41	3/21/2021 12:41
5		4 Manager	Table	Table	managertable@manag	https://atlas.noit.eu/storage/av	3/21/2021 12:41	3/21/2021 12:41
6								
7								
8								

Когато натиснем върху даден ред от таблицата, на екрана ни изкача модален прозорец, съдържащ данните от избрания от нас ред или от избрания потребител. В този прозорец, администраторите могат да променят данните на потребителя или да преглеждат съответния потребител. Полетата са следните:

- First Name (име)
- Middle Name (презиме)
- Last Name (фамилно име)
- Email (имейл адрес)
- City (град)
- Country (държава)
- Description (описание на потребителя)

### User



**FIRST NAME**

**MIDDLE NAME**

**LAST NAME**

**EMAIL**

**CITY**

**COUNTRY**

**CREATED**

**UPDATED**

**DESCRIPTION**

Update

Delete

+ ADD ROLE

	Name	Description
1	admin	Admin is a role, which is set on the main user. He has all permissions and can manage anything.

След секцията с полетата, забелязваме два бутона за промяната на данните на потребителя или за изтриването му. При натискане на някой от бутоните администраторът изтрива или променя данните след като потвърди запитването, което цели блокиране на неволното натискане.

Are you sure you want to change the data?

OK

Cancel

Are you sure you want to delete the data?

OK

Cancel

В най-долната част на модалния прозорец се намира таблицата с ролите на потребителя.

<div> <input type="text" value="Search Role"/> <input type="button" value="+ ADD ROLE"/> </div>	
Name	Description
1 admin	Admin is a role, which is set on the main user. He has all permissions and can manage anything.

Таблицата съдържа две колони: Name (име на ролята), Description (описание на ролята). Таблицата може да бъде сортирана според колоните си. В горната ѝ дясна част може да бъде намерен бутон „Add Role” или добави роля, който отваря втори модален прозорец. Този прозорец съдържа поле със списък на всичките 32 роли в нашата система. Всяка една роля може да бъде добавена на избрания потребител.

Role

admin

Press enter to select

ess

manager\_chart

manager\_chart\_slots\_by\_processing

manager\_chart\_slots\_by\_adc\_activity

manager\_chart\_slots\_by\_job\_type

manager\_chart\_slots\_vs\_n-cores

Add

Той може да управлява всичко, свързано с таблиците на диаграмите в уеб приложението. Ако даден потребител притежава ролята “manager\_chart”, “manager\_table” или “admin”, той може да види секцията с изброените BigPanDA и ATLAS диаграми в страничното навигиращо меню.

Когато потребителят избере някоя от диаграмите от списъка, на екрана се показва секция със снимката на диаграмата, която автоматично се взима от BigPanDA и се представя на потребителя в реално време от уеб приложението. Всяка една диаграма от нашето приложение, има собствена информация и значение. Тази информация дава ясна представа за това как и къде се използва, откъде се взимат данните и за колко време тези данни се използват. Всяка една от информациите ни е предоставена от учените и физиците на Европейска организация за ядрени изследвания (ЦЕРН). Тази информация, която предоставяме в нашето приложение е кратка, обобщена и лесна за разбиране. След като заредят основните данни на голямата диаграма от снимката, уеб приложението разделя тази „Stacked Bar” диаграма на отделни секции(сортирания) с динамични и анимирани поддиаграми, които имат за цел да предоставят на потребителя най-точно и най-подробно цялата налична информация.



След като потребителят кликне върху едно от сортиранията, той е навигиран до друга страница, представяща уголемен размер на цялата диаграма и под нея съответните процеси. За да види съответно едно от сортиранията и броя процеси през отделните часове, потребителят, чрез натискане, ще бъде въведен в динамична диаграма.

Отгоре на страницата е показана една от диаграмите („Slots by Processing”), която показва ресурсите, отделени за обработката на 2-та типа данни в ATLAS:

- обработка на реални данни от детектора (Данни);
- генериране на данни от теоретични модели (**MC - Monte Carlo**).

Веригата на обработка на данни в двата случая е следната (Под верига се разбира, че всяка една стъпка е вид обработка на данни и резултатите (изходящите данни) на дадена стъпка са входящи данни за следващата стъпка):

Данни:

- **Реконструкция (T0 Processing)** - Преработка (Data Processing) - Отсяване (Group Production) - Анализ (Analysis)

- **MC: Генериране (MC Generation)** - Симулация (MC Simulation) - Реконструкция (MC Reconstruction) - Отсяване (Group Production) - Анализ (Analysis)

- **T0 Processing** - Това е първоначалната обработка на данни от експеримента. Реконструират се частици и събития от данни от детекторите. Прави се само веднъж, веднага след като се случат събитията в експеримента и то в рамките на 48 часа от момента на сблъсъка.

- **Data Processing** - Прави се същата обработка на данни като T0, но само ако има подобрение в софтуера за обработка на данните - по-добре измервани пространствени/геометрични параметри, по-добро калибриране на детектора, по-добри алгоритми, изчистване на бъгове в софтуера и т.н.

- **Group Production** - Целта на тази стъпка е да могат да се намалят данните за даден физически анализ - намиране на Хигс бозон, или суперсиметрии, или черни дупки - до обем информация, която човек да може да си качи на лаптопа (гигабайти). Това става като се създават специални групи от файлове за отделните групи от анализи, като във всяка група се оставя само минималната информация която е необходима за дадения анализ:

- махат се събития/сблъсъци, в които няма това, което търсим (ако знаем, че в събитията които ще ни трябват за анализа трябва да има един електрон с енергия над определено ниво - махам всички събития в които няма такъв електрон).

- махат се всички (C++) обекти, които не са необходими (ако за анализа не са необходими електрони - махат се всички електрони).



- махат се всички свойства, които не са необходими (ако не ни интересува скоростта на електрона - махаме скоростта).

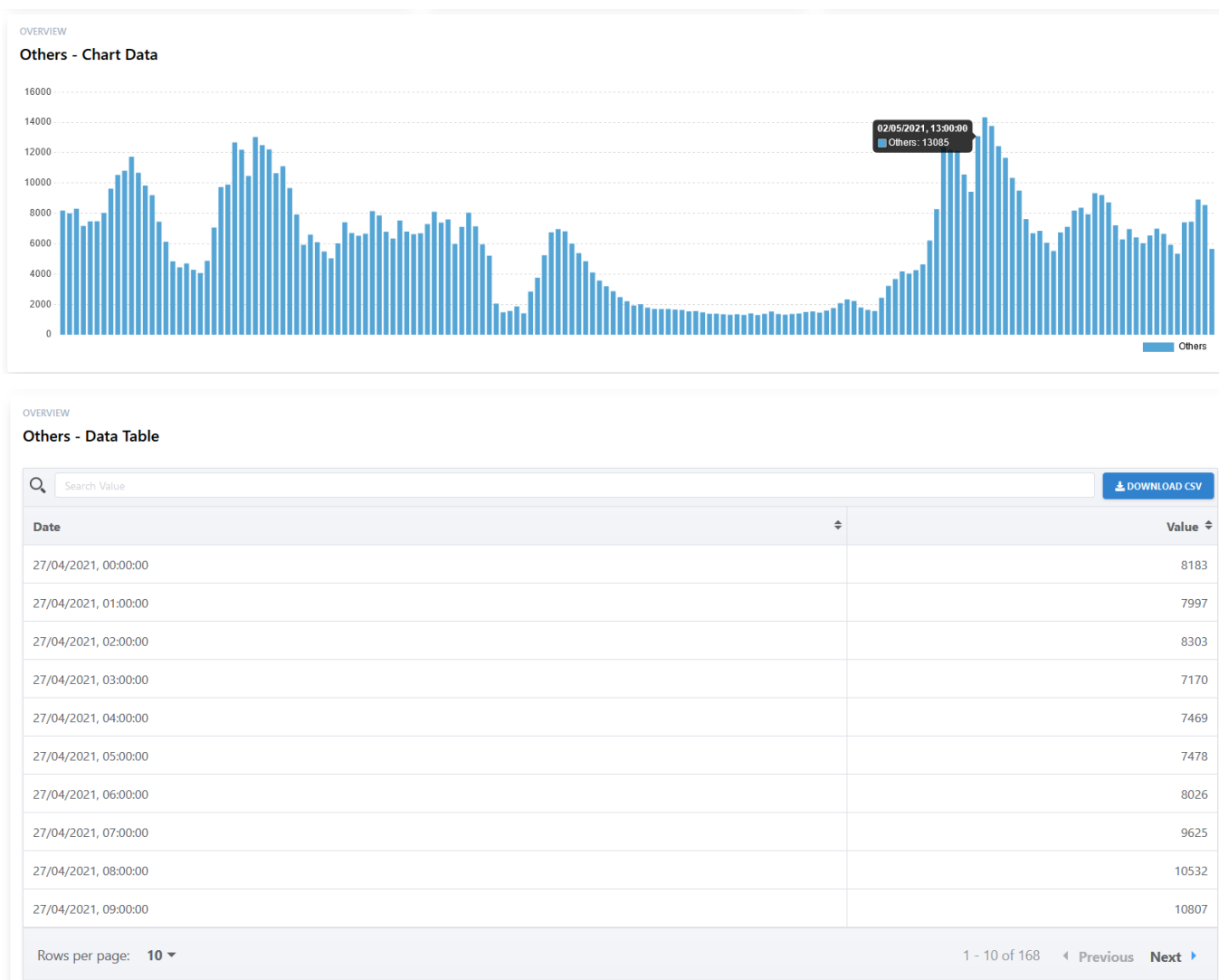
- **Analysis** - Реалната обработка на данните от физиците - в тази част се правят всички графики и плотове, които се показват по конференции и срещи.

- **MC Generation** - Генериране на събития. От теорията се знае с каква вероятност при сблъсък на два протона какво ще се получи. С генерирането от вероятности се получават събития. Изходящите данни са списъци с частици, с техния импулс и енергия.

- **MC Simulation** - Това е симулиране на преминаването на частиците през детектора. Знае се, че сблъсъкът на двата протона е станал в центъра на детектора, от там се знае каква частица се е родила и в каква посока е тръгнала, понеже знаеш импулса, а импулса има посока и знаеш в тази посока точно кой детектор се намира и как частицата ще взаимодейства с него и какъв сигнал детектора ще произведе от тази частица. След MC Simulation това, което имаме са детекторни сигнали. Изходящите данни от MC Simulation са същия формат, както и данните от детектора.

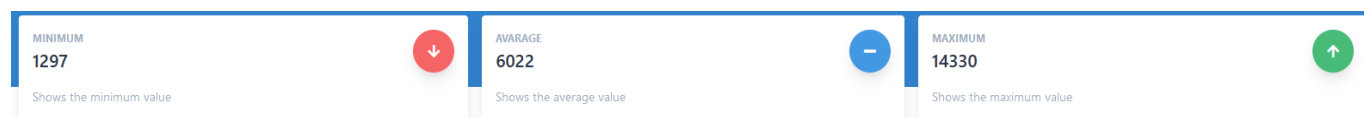
- **MC Reconstruction** - реконструират се частици и събития от детекторни сигнали.

При натискане на някой от следните “бутони” (Пример: Analysis, MC Reconstruction, Group Production или друга) приложението стартира екран с интерактивна диаграма или таблица ако потребителят притежава съответната роля. Тази диаграма дава точни, ясни и актуални данни, които се взимат под формата на JSON. Този JSON е предоставен от ЦЕРН. Те изпращат така наречените сурови данни (JSON), които ние използваме за създаването на тези интерактивни диаграми. Диаграмите, които ние използваме се наричат “Barchart”. Над всяко едно стълбче от диаграмата се виждат точните данни за всеки ден, които също са изобразени в диаграмата. Опитавме се да създадем лесно четими диаграми, за да може всеки потребител да намира всички данни, от които се нуждае без много усилия.



Когато сме в една от интерактивните диаграми, в горната част на нашето приложение виждаме три секции, които дават следната информация за тази диаграма:

- **Maximum** – най-високата стойност от съответната диаграма
- **Minimum** – най-ниската стойност от съответната диаграма
- **Average** – средната стойност от съответната диаграма



**4.8 Заключение:** В заключение можем да кажем, че “Atlas Service Management” е една удобна и лесна система за използване от всеки един потребител, независимо дали е работещ учен, ентузиаст физик или обикновен човек. Приложението, което сме създали с безвъзмездното предоставяне на „сурови“ данни от представител на експеримента ATLAS, е идеален инструмент за извличането на точни, лесно достъпни данни за конференции и събития, свързани с науката, като се показват всички процеси от експеримента ATLAS в ЦЕРН по един мобилен и иновативен начин.

