База данных состоит из 10 таблиц, которые условно можно разделить на три логические части: данные («Track», «Actor», «Tag»), связи («TrackActor», «TrackTag», «ActorTag»), пользователь («User», «UserTrack», «UserActor», «UserTag»).

Н.Н. Коваленко (УО «ГГУ им. Ф. Скорины», Гомель)
Науч. рук. **А.И. Кучеров,** старший преподаватель

КОНФИГУРАЦИЯ ПРОЕКТА МОНИТОРИНГА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ ЗАДАНИЙ УНИВЕРСИТЕТА НА NODE JS

Node JS – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Целью данного проекта являлось написание проекта, который упорядочит взаимодействие между обучающимися и преподавателями, Приложение позволяет:

1 Вести учёт лабораторных, практических и прочих работ.
2 Генерировать различные списки и отчёты об успеваемости.
3 Занимать очередь на сдачу какой-либо работы онлайн.
4 Записываться в группы к определённым преподавателям.
5 Регистрировать новых преподавателей.
6 Регистрировать новых обучающихся.

Для реализации вышеописанного функционала потребовалось сконфигурировать сервер на Node JS с использованием библиотек Express JS и Sequelize JS. Sequalize JS выступает в качестве объектно-реляционного преобразователя. Он связывает поля классов, описанных на ECMA6Script, с реальными таблицами базы данных. Благодаря этому на сервере мы может обращаться к таблицам и информации в них с помощью методов list(), save(), update() и т. д.

Библиотека Express JS в приложении служит помощником в предоставлении серверного API. Как только пошлём подобный запрос на сервер, на сервере на него отреагирует соответствующий обработчик, написанный с помощью Express JS. Он выглядит примерно следующим образом:

```
app.route('/badStudents', function(req, res) {
 let badStudents = Student.list { student =>
  where: {
   mark: student.mark < 6
  }
 }
})
```

Выбираются студенты с оценкой ниже 6. Так же в приложении используется библиотека Epilogue JS, которая в связке с Sequelize и Express JS даёт возможность получать список всех студентов, сохранять, удалять и обновлять студентов, не прибегая при этом к написанию обработчиков. Достаточно всего нескольких строк кода:

```
epilogue.resource({
 model: db.Student,
 endpoints: [
 '/students',
 '/students/:id'
 ]
});
```

При отправке GET запроса /api/students, получим список студентов. При отправке /api/students/1, получаем первого студента. При отправке PUT запроса /api/students сохраняем нового студента, а POST запроса /api/students/1 – можно обновить информацию о первом студенте.

После того, как серверная часть была сконфигурирована, нужно сконфигурировать клиентскую часть (интерфейс пользователя). Для конфигурации интерфейса использовался фреймворк React JS, предназначенный для написания «Single page» приложений. React JS – бесплатная JavaScript библиотека, предоставляющая интерфейс пользователя в виде react компонентов. В качестве компонента для приложения, был создан компонент ApplicationContext.

В приложение используется глобальный контекст, который виден в любом компоненте приложения. Он реализован с помощью модуля react-redux. Глобальный контекст содержит в себе объект currentUserInfo, который содержит информацию о текущем пользователе. Передвижение по single page приложению осуществляется с помощью модуля react-router, который предоставляет возможность отображать определённые страницы при переходе по определённым адресам, например:

```
<Router>
  <Route path=”/login” component={LoginPage} />
</Router>
```

Из кода следует, что проследовав по адресу http:/localhost:3000/login пользователь попадает на компонент LoginPage, который представляет из себя страницу входа в приложение. Роутер реализован в компоненте ReactRouter, который предоставляет весь список адресов.