

нечного автомата, четыре специализированных *AI*-агента и чат-интерфейс на *Blazor WebAssembly*. Система успешно протестирована на сценариях обучения по темам информационной безопасности с использованием реальных нормативных документов.

Разработанный интерактивный тренажер реализует диалоговый формат обучения, принципиально отличающийся от традиционных тестовых систем: при неверном ответе система не раскрывает правильного решения, а задает наводящий вопрос, побуждая обучаемого к самостоятельному нахождению ответа. Применение технологии *RAG* обеспечивает привязку всех генерируемых вопросов и оценок к актуальному содержимому загруженных документов. Реализация экзаменационного процесса в виде сериализуемого конечного автомата гарантирует предсказуемость педагогического сценария и масштабируемость системы. Гибкость настройки сценариев и поддержка локального развертывания делают тренажер применимым как в учебных заведениях, так и в организациях с ограничениями на передачу данных во внешние облачные сервисы.

Литература

1. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis, E. Perez, A. Piktus [et al.] // *Advances in Neural Information Processing Systems*. – 2020. – Vol. 33. – P. 9459–9474.
2. Microsoft. ASP.NET Core documentation. – URL: <https://learn.microsoft.com/en-us/aspnet/core/> (дата обращения: 15.03.2026).
3. Microsoft. Blazor WebAssembly documentation. – URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/> (дата обращения: 15.03.2026).
4. Microsoft. SK Process Framework Overview. – URL: <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/process/process-framework> (дата обращения: 20.03.2026).
5. Microsoft. Semantic Kernel Documentation. – URL: <https://learn.microsoft.com/en-us/semantic-kernel/> (дата обращения: 20.03.2026).

ИНТЕРАКТИВНАЯ СИСТЕМА ПЛАНИРОВАНИЯ И УЧЕТА ИНДИВИДУАЛЬНОЙ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЯ НА БАЗЕ ПЛАТФОРМЫ .NET 10

И. Ю. Тимошенко

Учреждение образования «Гомельский государственный технический
университет имени П. О. Сухого», Республика Беларусь

Научный руководитель О. Д. Асенчик

Описана интерактивная система планирования и учета индивидуальной учебной нагрузки преподавателя, предназначенная для замены электронных таблиц и снижения трудоемкости отчетности. Реализовано веб-приложение на платформе .NET 10 (Blazor Server) с гибридной моделью хранения данных (включая JSON) и механизмом «снимков» для сохранения архивной достоверности утвержденных планов. Представлены модуль импорта нагрузки из XLSX, алгоритм месячного распределения часов, учет фактического проведения и генерация отчетности; работоспособность решения подтверждена модульными и интеграционными тестами. Приложение подходит как для персонального использования, так и для внедрения в масштабе университета.

Ключевые слова: *Blazor Server, .NET 10, учебная нагрузка, информационная система, автоматизация документооборота, Entity Framework Core.*

В условиях современной цифровой трансформации образовательной среды задачи оперативного управления и контроля деятельности профессорско-препода-

вательского состава выходят на новый качественный уровень. Одной из наиболее трудоемких административных процедур на кафедре университета является ведение индивидуального плана преподавателя. Данный документ является не только инструментом планирования, но и основным юридическим основанием для подтверждения выполнения трудовых обязанностей сотрудником.

Традиционные методы ведения отчетности, базирующиеся на локальных файлах *MS Excel*, в современных реалиях демонстрируют ряд критических недостатков: отсутствие централизованного контроля версий, сложность объединения данных по кафедре, высокий риск нарушения целостности расчетных формул и значительные временные затраты на ручной перенос данных из общеуниверситетских ведомостей. Актуальность данной работы заключается в создании программного средства, которое позволит автоматизировать процесс заполнения и учета учебной нагрузки, обеспечивая при этом автоматическую валидацию.

При проектировании системы был сделан выбор в пользу платформы *.NET 10*. В качестве основного фреймворка выбрана технология *Blazor Server*. Данный выбор обоснован несколькими стратегическими преимуществами:

- использование языка *C#* как для бизнес-логики, так и для описания интерфейса позволяет избежать фрагментации кода и упрощает дальнейшую поддержку системы;
- приложение быстро реагирует на действия пользователя, не заставляя его ждать загрузки страницы или обновления контента [1];
- в архитектуре *Blazor Server* весь код выполняется на сервере, обновляя *DOM*-дерево через постоянное *WebSocket*-соединение, что важно для поддержания безопасности, так как чувствительные алгоритмы расчета нагрузки и параметры подключения к базе данных никогда не покидают периметр сервера;

Blazor Server поддерживает состояние сессии на сервере, что позволяет пользователю работать с таблицами ввода часов без потери промежуточных данных при кратковременных сбоях сети.

Одной из центральных проблем при проектировании систем учета нагрузки является изменчивость структуры данных. У разных преподавателей может быть разный набор дисциплин, видов работ и ежемесячных активностей.

В рамках исследования была разработана и внедрена гибридная модель хранения данных. База данных *SQLite* в связке с *Entity Framework Core* настроена таким образом, что основные сущности (пользователи, преподаватели, расписание) хранятся в реляционном виде, а сложные иерархические структуры (пакеты годовой нагрузки, помесечные расчеты, списки работ) – в формате *JSON*-строк внутри текстовых полей основной таблицы *Plans*. Реализация данного подхода стала возможной благодаря библиотеке *Newtonsoft.Json* и механизму *Value Conversions* в *EF Core*.

Особенностью системы является реализация механизма снимков для данных преподавателя. В типичных системах профиль пользователя связан с документом через идентификатор. Однако в контексте университетской отчетности это создает проблему: если преподаватель сменит должность или получит ученую степень в середине года, все его старые, уже утвержденные планы обновятся автоматически, что является недопустимым с точки зрения архивного дела.

Разработанная система при инициализации нового плана копирует всю информацию о преподавателе: Ф.И.О., кафедра, факультет, звание, должность и сохраняет ее как *JSON*-снимок внутри записи конкретного плана. Таким образом, документ становится автономным: он хранит информацию о статусе преподавателя именно на момент формирования отчета. При этом пользователь сохраняет возможность редактировать должность непосредственно в плане, если он работает по совместительству, не затрагивая свой основной профиль в системе.

Интерфейс планирования обеспечивает ввод данных по 14 видам учебной деятельности, разделенным на два семестра. К ним относятся:

- аудиторная нагрузка: лекции, лабораторные, практические и семинарские занятия;
- промежуточная и итоговая аттестация: консультации, экзамены, зачеты;
- контрольно-методическая работа: курсовое проектирование, курсовые работы, расчетно-графические работы, тесты;
- практики и дипломное проектирование: учебно-производственные практики, руководство дипломными проектами, рецензирование, государственные экзаменационные комиссии.

Для минимизации ошибок ручного ввода реализован модуль интеллектуального импорта, который анализирует структуру *XLSX*-файлов, позволяет пользователю выбрать конкретный лист книги и автоматически агрегирует данные по дисциплинам и группам, исключая дубликаты. Модуль позволяет импортировать разные форматы листов с годовой нагрузкой, автоматически определяя текущий формат по ключевым полям.

Центральным вычислительным модулем системы является сервис, реализующий алгоритм детерминированного распределения годовых часов по месячным интервалам. Данный алгоритм учитывает специфику академического календаря и сетку учебного расписания.

На основе этого формируется базис часов. Если суммарное количество часов по расписанию превышает лимит годового плана, алгоритм прекращает учет часов после достижения запланированного значения.

При вводе и корректировке данных о фактическом выполнении нагрузки, особенно часов по неаудиторной нагрузке, критически важным является необходимость их согласования с установленной годовой нагрузкой плановыми значениями. Особое внимание в работе уделено контролю целостности данных, который реализуется непосредственно в процессе ввода в режиме реального времени. Благодаря использованию функционала *.NET IO*, компоненты системы инициируют проверку сразу после изменения значения в любой ячейке. Применение системы проверок решает эту задачу, повышает эффективность и точность работы пользователя в программе, способствует повышению качества получаемой документации [2].

Математическая модель валидации проверяет выполнение условия:

$$\sum_{j=1}^m H_{\text{месяц } j} = H_{\text{год}},$$

где m – количество месяцев в семестре. При возникновении неравенства система вычисляет расхождение и выполняет следующие действия:

- визуально маркирует ячейки с ошибками;
- формирует список диагностических сообщений с указанием точного объема избыточных или недостающих часов;
- блокирует транзакцию сохранения в базу данных до момента достижения нулевого баланса.

Такой подход позволяет исключить формирование ошибочных отчетов, что является критически важным для проверки отчетов работниками учебного отдела университета.

Для заполнения блоков учебно-методической, научно-исследовательской, общественной и идеологической работы в систему интегрирован справочник. Преподаватель имеет возможность мгновенного добавления типовых формулировок из выпадающего меню-справочника с возможностью ручного редактирования, что сокращает время заполнения данных разделов.

Итоговым функционалом системы является модуль генерации отчетности на базе библиотеки *ClosedXML*. Техническая сложность реализации данного модуля заключалась в необходимости сохранения структуры оригинального университетского бланка. Результатом является файл формата *XLSX*, готовый к подаче на подпись заведующему кафедрой.

Универсальность разработанного программного комплекса заключается в его архитектурной гибкости, позволяющей использовать систему как в качестве локального инструмента персонального планирования, так и в роли централизованного университетского сервиса.

Для личного использования преимуществом является выбор СУБД *SQLite* и платформы *.NET*, что обеспечивает портативность: приложение может быть запущено на ПК преподавателя без необходимости настройки серверной инфраструктуры. Модули импорта и экспорта позволяют пользователю автономно обрабатывать вузовские ведомости и получать готовые отчеты, не дожидаясь внедрения системы на уровне всего учебного заведения.

Для внедрения в масштабе университета система обладает необходимым запасом масштабируемости и безопасности. Использование *ASP.NET Core Identity* обеспечивает разграничение прав доступа и изоляцию данных между пользователями. Наличие централизованных справочников дает администрации вуза возможность унифицировать формулировки отчетности для всех кафедр.

Надежность разработанного программного комплекса подтверждена комплексом из 18 модульных и интеграционных тестов на базе фреймворка *xUnit*.

Разработанное программное средство обеспечивает комплексную автоматизацию учета учебной нагрузки, устраняя основные недостатки традиционных методов планирования. Внедрение реактивного интерфейса *Blazor Server* в сочетании с алгоритмами распределения позволило создать надежный и удобный инструмент для профессорско-преподавательского состава. Результаты тестирования и апробации модуля импорта/экспорта подтвердили готовность системы к эксплуатации в условиях реального документооборота кафедры. Репозиторий проекта расположен по адресу <https://github.com/kussz/PersonalPlanShowcase>.

Л и т е р а т у р а

1. Илья Лебедев / Принципы реактивного программирования в веб-разработке. – URL: <https://tenchat.ru/media/2080930-printsipy-reaktivnogo-programmirovaniya-v-vebrazrabotke> (дата обращения: 02.04.2026).
2. Широченко, В. А. / Автоматизированная система расчета, распределения и анализа учебной нагрузки кафедры вуза / В. А. Широченко // Вестник Белорусско-Российского университета. – 2024. – № 3 (84). – URL: <https://cyberleninka.ru/article/n/avtomatizirovannaya-sistema-rascheta-raspredeleniya-i-analiza-uchebnoy-nagruzki-kafedry-vuza> (дата обращения: 03.04.2026).

СИСТЕМА АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ КАЧЕСТВА FDM-ПЕЧАТИ НА БАЗЕ КОМПЬЮТЕРНОГО ЗРЕНИЯ И АНАЛИЗА G-КОДА

Е. В. Леонович

*Учреждение образования «Гомельский государственный технический
университет имени П. О. Сухого», Республика Беларусь*

Научный руководитель В. В. Комраков

Рассмотрен процесс разработки программного обеспечения для автоматизированного контроля качества аддитивного производства методом FDM. Описаны алгоритмические