БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ

Ректор Белорусского
государственного университета

_____А.Д.Король

27 июня 2025 г.

Регистрационный № 3792/б.

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебная программа учреждения образования по учебной дисциплине для специальности:

6-05-0533-10 Информатика

2025 г.

Учебная программа составлена на основе ОСВО 6-05-0533-10-2023, учебных планов БГУ № 6-5.3-58/01, № 6-5.3-58/02, № 6-5.3-58/03, № 6-5.3-58/04, № 6-5.3-58/05 от 15.05.2023.

СОСТАВИТЕЛЬ:

О.Г.Казанцева, старший преподаватель кафедры многопроцессорных систем и сетей факультета прикладной математики и информатики Белорусского государственного университета

РЕЦЕНЗЕНТЫ:

Е.А.Левчук, доцент кафедры технологий программирования факультета прикладной математики и информатики Белорусского государственного университета, кандидат технических наук;

Н.В.Лапицкая, заведующий кафедрой программного обеспечения информационных технологий УО «Белорусский государственный университет информатики и радиоэлектроники», кандидат технических наук, доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой многопроцессорных систем и сетей БГУ
(протокол № 14 от 02.06.2025)

Научно-методическим советом БГУ
(протокол № 11 от 26.06.2025)

Заведующий кафедрой _____ И.Е.Андрушкевич

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Цели и задачи учебной дисциплины

Цель учебной дисциплины – обучение студентов современным методологиям и лучшим практикам управления IT-проектами, проектирования и разработки программного обеспечения, работы в команде и использования эффективных инструментов разработки, тестирования и развертывания приложений. В рамках курса у студентов формируются глубокие и систематизированные знания о полном жизненном цикле разработки программного обеспечения, включая этапы инициации, проектирования, разработки, тестирования, развертывания и поддержки. В основу курса положено развитие у студентов компетенций по выбору, проектированию и реализации архитектурных решений для создания масштабируемых, надёжных, поддерживаемых и высокопроизводительных распределенных систем и веб-приложений.

Задачи учебной дисциплины:

1. Сформировать системное понимание методологий и процессов разработки программного обеспечения, включая жизненный цикл ПО, стандарты ISO/IEC/IEEE, управление требованиями, проектную документацию и визуальное моделирование с использованием UML.
2. Развить навыки архитектурного проектирования и клиентской разработки, охватывающие принципы SOLID, DRY, DDD, стили распределённых систем (клиент-сервер, микросервисы, событийно-ориентированные), а также реализацию взаимодействий, маршрутизации, управления состоянием и безопасности.
3. Обучить современным подходам к проектированию пользовательских интерфейсов (UX/UI), включая адаптивность, доступность, инклюзивность и применение инструментов дизайна и фреймворков для создания эффективных клиентских приложений.
4. Сформировать практические навыки командной разработки и автоматизации процессов, включая работу с системами контроля версий (Git), инспекцию кода, тестирование, журналирование, а также освоение DevOps-практик, CI/CD, контейнеризации и инструментов автоматизации (Docker, Jenkins).

Место учебной дисциплины в системе подготовки специалиста с высшим образованием.

Учебная дисциплина относится к модулю «Программирование» государственного компонента.

Связи с другими учебными дисциплинами, включая учебные дисциплины компонента учреждения высшего образования, дисциплины специализации и др.

Учебная программа составлена с учетом межпредметных связей и программ по дисциплинам. Основой для изучения учебной дисциплины являются дисциплины государственного компонента «Основы и методологии программирования» и «Промышленное программирование» модуля «Программирование», «Операционные системы» и «Модели данных и СУБД»

модуля «Компьютерные системы», «Основы теоретической информатики» и «Алгоритмы и структуры данных» модуля «Дискретная математика и алгоритмы».

Знания, полученные в учебной дисциплине, используются при выполнении студентами курсовых проектов, курсовых и дипломных работ.

Требования к компетенциям

Освоение учебной дисциплины «Технологии программирования» должно обеспечить формирование следующих универсальных, базовых профессиональных компетенций*:*

Универсальные компетенции:

Владеть основами исследовательской деятельности, осуществлять поиск, анализ и синтез информации.

Решать стандартные задачи профессиональной деятельности на основе применения информационно-коммуникационных технологий.

Работать в команде, толерантно воспринимать социальные, этнические, конфессиональные, культурные и иные различия.

Быть способным к саморазвитию и совершенствованию в профессиональной деятельности.

Базовые профессиональные компетенции:

Применять при проектировании приложений такие парадигмы программирования как структурное, объектно-ориентированное и функциональное программирование

Разрабатывать программное обеспечение в интегрированных средах разработки

В результате освоения учебной дисциплины студент должен:

знать:

– основные методологии разработки ПО, включая Agile и жизненный цикл требований;

– базовые диаграммы UML и их применение в проетировании приложений;

– архитектурные стили распределённых систем;

– принципы проектирования гибких и масштабируемых систем;

– протоколы взаимодействия в распределенных системах и безопасность клиент-серверных решений;

– основы UX/UI-дизайна, адаптивной верстки и подходов Mobile First;

– архитектуру и особенности современных клиентских фреймворков;

– принципы работы с Git, Code Review, журналированием и тестированием (xUnit);

– концепции DevOps, CI/CD, контейнеризацию (Docker) и автоматизацию (Jenkins), этапы конвейера поставки;

уметь:

– разрабатывать проектную документацию и управлять требованиями;

– проектировать приложения с использованием UML и архитектурных паттернов;

– создавать и подключать базы данных, писать сложные SQL-запросы;
– применять принципы архитектурного проектирования;
– реализовывать взаимодействие сервисов через REST API;
– разрабатывать адаптивные и доступные интерфейсы с Flexbox и Grid;
– создавать клиентские приложения на современных фреймворках, включая маршрутизацию и обработку ошибок;
– использовать Git, проводить «code review», настраивать журналирование и тестирование;
– применять Docker и Jenkins для автоматизации процессов CI/CD;

иметь навык:

– участия в командной разработке по методологии Scrum и управления требованиями проекта;
– проектирования и реализации веб-приложений от идеи до MVP, включая UML-моделирование и работу с СУБД;
– оценки архитектурных решений и применения современных инструментов разработки (IDE, Git, Docker, Jenkins, Figma);
– автоматизации процессов CI/CD.

Структура учебной дисциплины

Дисциплина изучается в 5 семестре. В соответствии с учебным планом всего на изучение учебной дисциплины «Технологии программирования» отведено **для очной формы** получения высшего образования – 108 часов, в том числе 68 аудиторных часов, лекции – 34 часа, лабораторные занятия – 34 часа. **Из них:**

Лекции – 34 часов, лабораторные занятия – 30 часов, управляемая самостоятельная работа (УСР) – 4 часов.

Трудоемкость учебной дисциплины составляет 3 зачетные единицы.

Форма промежуточной аттестации – зачет.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Раздел 1 Инициализация и проектирование приложений

Тема 1.1 Методологии разработки ПО

Введение в управление ИТ-проектами. Эволюция методологий жизненного цикла разработки программного обеспечения. Процессы жизненного цикла согласно ISO/IEC/IEEE 12207:2017.

Гибкие методологии разработки программного обеспечения. Базовые термины проектного управления. Принципы и ценности Agile. Обзор популярных гибких методологий на примере RUP Scrum, XP, Lean Kanban и др. Применение Scrum для управления проектами.

Тема 1.2 Инициация проекта, выявление требований, общее планирование проекта

Инициация проекта. Инструменты командной разработки проекта.

Виды требований к программному обеспечению. (бизнес-требования, пользовательские, функциональные, нефункциональные).

Выявление требований. Анализ требований. Спецификация требований. Валидация требований. Верификация требований. Трассировка требований. Управление требованиями.

Документирование проекта: «Устав проекта», «Техническое задание», «Реестр рисков» и «План управления рисками», «Стратегии работы с заинтересованными сторонами», «План управления коммуникациями», «План управления персоналом», «План управления качеством», «Руководство пользователя приложения».

Тема 1.3 Проектирование приложения с помощью UML (Unified Modeling Language)

Структура UML. Пакеты в языке UML. Основные пакеты метамодели UML. Описание метамодели языка UML. Диаграммы UML. Диаграмма вариантов использования (Use Case). Диаграмма классов. Диаграмма состояний. Диаграмма деятельности (Activity). Диаграмма последовательности. Диаграмма кооперации. Диаграммы компонентов и развертывания.

Тема 1.4 Управление данными

История развития СУБД. Управление данными. Основные виды СУБД. Типы данных. Принципы управления данными. Современные тенденции в управлении данными.

Создание базы данных, таблиц, вставка строк и удаление. Выборки данных и параметрические запросы. Индексы. Триггеры.

Подключение к базе данных из приложения.

Раздел 2 Проектирование и разработка клиентского приложения

Тема 2.1 Проектирование пользовательского интерфейса

Основные принципы и инструменты дизайна и верстки веб-приложений.

Цветовая палитра и типографика.

Принципы проектирования UX (User Experience) и UI (User Interface). Основные элементы UX/UI дизайна. Этапы проектирования. Проектирование с учетом доступности и инклюзивности.

Инструменты для дизайна: Sketch, Figma, Adobe XD.

Основы верстки веб-приложений. Принципы адаптивного дизайна (Responsive design). Современные методы верстки для создания гибких и адаптивных макетов: Flexbox, Grid layout.

Подходы Atomic Design Mobile first, Progressive Enhancement.

Тема 2.2 Фреймворки для разработки клиентских веб-приложений

Формат обмена данными JSON.

Основные особенности и сравнительный анализ фреймворков React, Vue, Angular, Svelte. Основные концепции: маршрутизация, обработка запросов, валидация данных, оптимизация производительности. Взаимодействие с API сервера. Обработка ошибок и управление состоянием.

Интернационализация и локализация. Локализация интерфейса и файлы локализации.

Раздел 3. Инструменты для разработки приложений

Тема 3.1 Версионный контроль

Отслеживание изменений кода. Эволюция и знаковые систем контроля версий. Обзор и введение Git. Инспекция кода (code review). Рабочие процессы при версионировании (workflow).

Система контроля версий git и сборка приложений. Введение в системы контроля версий (СКВ). Типы СКВ. Распределенная система контроля версий git. Установка git. Создание и инициализация репозитория. Клонирование репозитория. Запись изменений в репозиторий. Внешние репозитории. Публикация изменений. Управление ветками. Консольный git клиент. Графические клиентские приложения. управления версиями.

Файл Makefile. Утилиты make, cmake. Современные решения для автоматической сборки и распространения.

Тема 3.2 Журналирование событий, анализ и тестирование кода

Журналирование событий. Уровни журналирования, форматы сообщений, направления вывода сообщений.

Инструменты анализа кода. Документирование кода.

Виды тестирования ПО. Модульное тестирование, семейство xUnit.

Особенности тестировании клиентской части приложения.

Тема 3.3 Практики DevOps. Непрерывная интеграция (CI). Непрерывная поставка и развертывание (CD)

Типовые проблемы при совместной разработке. Конвейер (pipeline) и основные этапы. Лучшие практики. Популярные инструменты. Виды артефактов.

Введение в Docker: настройка контейнеризации через Docker, запуск приложений в контейнерах, построение образов. Введение в Jenkins.

Основы CI/CD. Определение, цели и задачи. Конвейер поставки. Основные трудности. Требования к приложениям. Версионирование артефактов. Решение проблем с изменением окружений. Обновление и откат приложений. Развёртывание приложений и доступ через сервисы.

Раздел 4 Архитектуры программного обеспечения

Тема 4.1 Распределенные системы и их архитектурные стили

Определения распределенных систем. Области применения и примеры. Типы распределенных систем, их отличия и особенности.

Концептуальная архитектура. Архитектурный стиль. Пилотная архитектура. Модуль. Компонент. Фреймворк.

Архитектурные стили: «Клиент-серверная архитектура» (Client-server architecture), «Чистая архитектура» (Clean Architecture), «Ресурсные архитектуры» (Resource-Oriented Architectures, ROA), «Архитектура одноранговой сети» (Peer-to-peer architecture), «Архитектура с брокером» (Broker architecture) и ее частный случай «Архитектура публикация-подписка» (Publish-subscribe), «Архитектура с реплицированием» (Replication architecture), «Архитектура с распределенными объектами» (Distributed object architecture), «Архитектура с распределенными агентами» (Distributed agent architecture).

Тема 4.2 Архитектуры, концепции и принципы создания гибких, масштабируемых и поддерживаемых систем

Виды архитектур программного обеспечения: «Многослойная архитектура» (Layered architecture), «Многоуровневая архитектура» (Tiered architecture), «Сервис-ориентированная архитектура» (Service oriented architecture – SOA), «Микросервисная архитектура» (Microservice architecture), «Событийно-ориентированная архитектура» (Event-driven architecture), «Архитектура на основе компонентов» (Component-Based Architecture).

Системная (программная) архитектура распределенных систем: централизованные, гибридные архитектуры.

Аспектно-ориентированное проектирование (Aspect-oriented programming) и предметно-ориентированное проектирование (Domain-driven design – DDD).

Принципы объектно-ориентированного проектирования SOLID. Принципы DRY, WET, KISS, YAGNI, POLA, SoC и др..

Тема 4.3 Взаимодействие между процессами в распределенных системах

Возможные разновидности взаимодействий. Передача сообщений между парой процессов. Схема «запрос-ответ» и удаленные вызовы процедур (RPC).

Многоуровневые протоколы.

Протокол прикладного уровня HTTP. Эволюция протокола HTTP. Структура HTTP-запроса и ответа. URI, URL, URN.

Веб-сервисы. Ресурсные архитектуры REST (Representational State Transfer) и WebDAV (Web Distributed Authoring and Versioning). Сравнение архитектуры REST с RPC.

Протокол обмена структурированными сообщениями SOAP (Simple Object Access Protocol). Сравнение подходов SOAP и REST.

Групповые взаимодействия. Надежная рассылка сообщений в группе. Масштабируемые подходы к распространению информации.

Непрямое взаимодействие. Очередь сообщений. Издатель-подписчик. Распределенная общая память. Пространство кортежей.

Раздел 5 Разработка клиент-серверных приложений

Тема 5.1 Именование и поиск ресурсов в системе

Имена, идентификаторы и адреса. Виды имен, схемы именования, разрешение имени.

Понятие хостинга, домена. Иерархический подход со структурированным пространством имен. DNS (Domain Name System). DHT (Distributed Hash Table). Децентрализованный подход с плоским пространством имен. Именование на основе атрибутов. Протокол LDAP (Lightweight directory access protocol).

Тема 5.2 Безопасность приложений

Требования к безопасности в распределенных системах. Базовые техники и механизмы. Шифрование.

Аутентификация. Авторизация. Способы аутентификации в веб-приложениях. Управление доступом на основе ролей (RBAC) и атрибутов (ABAC).

Куки и токены. Виды токенов: OAuth, JWT (JSON Web Token), SWT (Simple Web Token), SAML (Security Assertion Markup Language). Хеширование пароля.

Цифровая подпись. Управление ключами.

Тема 5.3 Масштабирование и кэширование приложений

Масштабируемость и масштабирование. Техники масштабирования и связанные задачи. Репликация stateless-сервиса и балансировка нагрузки.

Разбиение stateful-сервиса по данным (шардинг).

Кэширование. Использование кэширования для оптимальной производительности API и веб-приложений.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Очная (дневная) форма получения высшего образования с применением дистанционных образовательных технологий (ДОТ)

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Количество часов УСР	Форма контроля
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1	**Инициализация и проектирование приложений**	**10**			**8**		**2**	
1.1	Методологии разработки ПО	2						Экспресс-опрос
1.2	Инициация проекта, выявление требований, общее планирование проекта	4			4			Экспресс-опрос, отчет
1.3	Проектирование приложения с помощью UML (Unified Modeling Language)	2			2		2	Контрольная работа №1
1.4	Управление данными	2			2			Экспресс-опрос, отчет
2	**Проектирование и разработка клиентского приложения**	**6**			**6**			
2.1	Проектирование пользовательского интерфейса	2			2			Экспресс-опрос, отчет
2.2	Фреймворки для разработки клиентских веб-приложений	4			4			Контрольная работа №2

3	**Инструменты для разработки приложений**	**6**			**4**		**2**	
3.1	Версионный контроль	1						
3.2	Журналирование событий, анализ и тестирование кода	3			2			Отчет
3.3	Практики DevOps. Непрерывная интеграция (CI). Непрерывная поставка и развертывание (CD)	2			2		2	Контрольная работа №3
4	**Архитектуры программного обеспечения**	**6**			**6**			
3.1	Распределенные системы и их архитектурные стили	2						Экспресс-опрос
3.2	Архитектуры, концепции и принципы создания гибких, масштабируемых и поддерживаемых систем	2			2			Экспресс-опрос, отчет
3.3	Взаимодействие между процессами в распределенных системах	2			4			Экспресс-опрос, отчет
5	**Разработка клиент-серверных приложений**	**6**			**6**			
5.1	Именование и поиск ресурсов в системе	2						Экспресс-опрос
5.2	Безопасность приложений	2			4			Контрольная работа №4
5.3	Масштабирование и кэширование приложений	2			2			Экспресс-опрос, отчет

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Основная литература

1. Гамма Э. Паттерны объектно-ориентированного проектирования / Э. Гамма [и др. ; пер. с англ. А. Слинкин]. - Санкт-Петербург ; Москва ; Минск : Питер, 2023. - 446 с. - URL: https://ibooks.ru/bookshelf/371734.

2. Маран, М. М. Программная инженерия : учебное пособие / М. М. Маран. - Изд. 3-е, стер. - Санкт-Петербург ; Москва ; Краснодар : Лань, 2022. - 194 с. - URL: https://reader.lanbook.com/book/189470#2.

3. Стин ван М. Распределенные системы / пер. с англ. В. А. Яроцкого / М. Стин ван, Э. С. Таненбаум – М.: ДМК Пресс, 2021. - 583 с.

4. Хориков, В. Принципы юнит-тестирования / Владимир Хориков ; [пер. с англ. Е. Матвеев]. - Санкт-Петербург ; Москва ; Минск : Питер, 2024. - 316 с. - https://ibooks.ru/reading.php?short=1&productid=373514.

Дополнительная литература

1. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. - Москва: Издательство Юрайт, 2022. - 432 с.

2. Попова, Ю. Б. Тестирование и отладка программного обеспечения: пособие / Ю. Б. Попова. - Минск : БНТУ, 2020. - 66 с.

3. Фримен, Э. Head First. Паттерны проектирования = Head First. Design Patterns / Эрик Фримен, Элизабет Робсон при участии Кэтти Сьерра и Берта Бейтса ; [пер. с англ. Е. Матвеева]. - Обновленное юбилейное изд. - Санкт-Петербург [и др.]: Питер, 2019. - 651 с. - URL: https://ibooks.ru/bookshelf/377150.

4. 4 .Чакон, С. Git для профессионального программиста / Чакон С., Штрауб Б.. – СПб.: Питер. Библиотека программиста, 2022. - 496 с.

5. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Пер. с англ. - СПб: Символ-Плюс, 2007. - 624 с.

6. Балашов А.И., Рогова Е.М., Тихонова М.В., Рогова Е.М. Управление проектами:учебник. - М.: Юрайт, 2014. - 383 с.

7. Бахтизин В.В., Глухова Л.А. Метрология, стандартизация и сертификация в информационных технологиях. Учебное пособие. - В 2-х частях. - Минск: БГУИР, 2016.

8. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. Пер. с англ. - М.: ДМК Пресс, 2014. - 496 с.

9. Вигерс К., Битти Д. Разработка требований к программному обеспечению, издание третье. - М: Русская редакция, 2018. - 716 с.

10. Лаврищева Е. М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования / Е. М. Лаврищева. - К.:Наук. думка, 2013. - 283 с.

11. Ларман Крэг. Применение UML и шаблонов проектирования. 2-е издание. Пер. с англ. - М.: Издательский дом «Вильямс», 2007. - 727 с.

12. Макконнелл С. Совершенный код. Мастер класс / Пер. с англ. - М.: Издательство «Русская редакция», СПб. : Питер, 2016. - 896 с.
13. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. - СПб.: Питер, 2018. - 352 с. - URL: https://ibooks.ru/bookshelf/361841.
14. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. - СПб.: Питер, 2013. - 464 с.
15. Ошероув Р. Искусство автономного тестирования с примерами на C#. 2-е издание / пер. с англ. Слинкин А. А. - М.: ДМК Пресс, 2014. - 360 с.

Электронные ресурсы

1. Образовательный портал БГУ [Электронный ресурс]. – Режим доступа: https://edufpmi.bsu.by/course/view.php?id=972. – Дата доступа: 15.04.2025.

Перечень рекомендуемых средств диагностики и методика формирования итоговой отметки

Объектом диагностики компетенций студентов являются знания, умения, полученные ими в результате изучения учебной дисциплины. Выявление учебных достижений студентов осуществляется с помощью мероприятий текущей и промежуточной аттестации.

Для диагностики компетенций могут использоваться следующие средства текущей аттестации:

1. контрольная работа;
2. экспресс-опрос на аудиторных занятиях;
3. отчет о выполнении лабораторных работ;

Формой промежуточной аттестации по дисциплине «Технологии программирования» учебным планом предусмотрен зачет.

Примерный перечень заданий для управляемой самостоятельной работы

Тема 1.3 Проектирование приложения с помощью UML (Unified Modeling Language) (2 ч).

Для заданного проекта и на основе пользовательских требований выполнить на языке UML разработку диаграмм состояний, последовательности, деятельности.

Тема задания: Проектирование приложения с помощью UML (Unified Modeling Language).

Форма контроля – контрольная работа.

Тема 3.3 Практики DevOps. Непрерывная интеграция (CI). Непрерывная поставка и развертывание (CD) (2 ч).

Разработать скрипты для создания контейнеров для серверной и клиентской частей приложения.

Тема задания: Основы CI/CD. Настройка контейнеризации через Docker.

Форма контроля – контрольная работа.

Примерный перечень лабораторных занятий

1. Лабораторное занятие №1-2. Инициализация проекта и выявление требований.
2. Лабораторное занятие №3. Проектирование приложения с помощью UML.
3. Лабораторное занятие №4. Управление данными. Разработка API сервера.
4. Лабораторное занятие №5. Проектирование пользовательского интерфейса.
5. Лабораторное занятие №6-7. Разработка клиентского приложения.
6. Лабораторное занятие №8. Организация эффективной работы с кодом: версионный контроль, логирование, тестирование, инструменты анализа кода. Документирование кода.
7. Лабораторное занятие №9. Основы CI/CD. Настройка контейнеризации через Docker.
8. Лабораторное занятие №10-12. Разработка RESTful приложения. Протокол HTTP и веб-сервисы.
9. Лабораторное занятие №13-14. Управление доступом на основе ролей. Безопасность приложения.
10. Лабораторное занятие №15. Масштабирование и кэширование приложений.

Рекомендуемая тематика контрольных работ

1. Контрольная работа № 1 «Методологии, требования и UML».
2. Контрольная работа № 2 «Проектирование клиентского приложения».
3. Контрольная работа № 3. «Инструменты разработки и DevOps».
4. Контрольная работа № 4. «Архитектура и клиент-серверные взаимодействия».

Примерные варианты контрольных работ

Контрольная работа № 1

Цель: проверить понимание жизненного цикла разработки ПО, гибких методологий, управления требованиями и основ UML.

Задание:

1) Составить «Устава проекта» и «Техническое задание» на основе кейса.
2) Выполнить классификацию требований (бизнес, функциональные, нефункциональные).
3) Определить Scrum board для проекта.

4) Разработать UML-диаграмму Use Case, и разработать пользовательские истории и сценарии вариантов использования.

Контрольная работа № 2

Цель: оценить навыки проектирования интерфейса и разработки клиентской части с использованием фреймворков.

Задание:

1) выполнить роектирование UX/UI макета в Figma или Sketch с учётом доступности;

2) Выполнить верстку адаптивного интерфейса с использованием Flexbox и Grid;

3) реализовать простого клиентского приложения на выбранном фреймворке (React/Angular);

4) настроить маршрутизацию, локализацию и обработку ошибок.

Контрольная работа № 3

Цель: проверить владение системами контроля версий, тестированием, сборкой и CI/CD.

Задание:

1) использовать работу с Git в командной разработке проекта. Разработку функциональных требований проводить в отдельных ветках. Выполнить Pull Request, Code Review;

2) реализовать модульные тесты (xUnit) и журналирование событий для серверной и клиентской части;

3) настроить для проекта CI/CD пайплайна с Docker и Jenkins: сборка, тестирование, деплой.

Контрольная работа № 4

Цель: оценить понимание архитектурных стилей, взаимодействия в распределённых системах и обеспечения безопасности.

Задание:

1) выполнить выбор архитектуры приложения, проведя анализ клиент-серверной, микросервисной, событийно-ориентированной архитектуры;

2) выполнить проектирование REST API приложение и реализовать взаимодействие серверной и клиентской часть приложения через HTTP.

3) провести анализ безопасности приложения: авторизация (JWT, OAuth), шифрование, управление доступом.

Описание инновационных подходов и методов к преподаванию учебной дисциплины

Преподавание дисциплины «Технологии программирования» строится на принципах активного обучения, командной инженерной практики и гибкой методологии Scrum. Основная цель – не только передача знаний, но и формирование устойчивых профессиональных навыков через имитацию реальных процессов разработки.

Все задания на курсе связаны между собой и имеют целью сформировать глубокие и систематизированные знания о полном жизненном цикле разработки программного обеспечения, включая этапы инициации, проектирования, разработки, тестирования, развертывания и поддержки.

Для выполнения лабораторных работ студенты всего потока разбиваются на кросс-функциональные команды по 4 человека: 2 backend-разработчика и 2 frontend-разработчика. Также в командах они выполняют следующие роли при работе над совместных проектом: Product Owner, Teamlead, DevOps, Дизайнер, Технический писатель.

Команда работает над одним проектом, но при этом, в процессе работы в рамках курса, участники группы работают как над общими, так и специфическими учебными заданиями. Каждая команда разрабатывает собственный минимально жизнеспособный продукт (MVP-продукт), проходя полный цикл: от инициации проекта до CI/CD и масштабирования.

Работа ведется на основе Scrum с использованием лучших практик управления, проектирования и разработки программного обеспечения.

Работа организована в спринтах (2 недели), с обязательными артефактами Scrum: Product Backlog, Sprint Planning, Daily Stand-ups (еженедельная отчетность), Sprint Review и Retrospective, Scrum Metrics.

Преподаватель выступает как Scrum-мастер и технический ментор, направляя команды, позволяя командам принимать их внутренние решения.

При организации образовательного процесса используются следующие методы:

- *метод учебной дискуссии*, который предполагает участие студентов в целенаправленном обмене мнениями, идеями для предъявления и/или согласования существующих позиций по определенной проблеме. Использование метода обеспечивает появление нового уровня понимания изучаемой темы, применение знаний (теорий, концепций) при решении проблем, определение способов их решения. Это способствует развитию критического мышления и аргументации технических решений.

- *метод группового обучения*, который представляет собой форму организации учебно-познавательной деятельности обучающихся, предполагающую функционирование малых групп: работу в Scrum-командах, включающих frontend и backend разработчиков.

- *проектно-ориентированное обучение:* курс построен вокруг реальных кейсов и проектных задач, охватывающих все этапы разработки: архитектура, UX/UI, взаимодействие с API, безопасность, DevOps.

Студенты создают проектную документацию (Устав проекта, ТЗ, диаграммы UML, планы управления рисками и др.), что формирует навыки инженерного мышления.

Командная разработка проектов ведется с использованием профессиональных инструментов: GitHub, Figma, Docker, Jenkins, Postman, VS Code, Jira/Trello.

Все задания выполняются в репозиториях с версионным контролем, с обязательным «code review» и CI-пайплайном.

Студенты осваивают DevOps-практики, включая автоматическую сборку, тестирование и развертывание приложений.

При проектировании приложения применяются UML-диаграммы, mind maps, архитектурные схемы, что развивает системное мышление.

Задания включают сравнительный анализ архитектурных стилей, визуальное моделирование взаимодействий и проектирование REST API.

- *метод открытой коммуникация и peer-to-peer обучение* – через ретроспективу, «code review» и взаимное наставничество внутри команд студенты учатся давать конструктивную обратную связь, аргументировать технические решения и работать в условиях распределённой ответственности.

Преподаватель поощряет взаимное обучение, где сильные участники помогают менее опытным, формируя культуру инженерной поддержки.

Методические рекомендации по организации самостоятельной работы

В качестве технических средств для организации самостоятельной работы по учебной дисциплине «Технологии программирования» используется «Образовательный портал ФПМИ БГУ». На курсе дисциплины https://edufpmi.bsu.by/course/view.php?id=972 представлены методические материалы: презентации лекций, методические указания к лабораторным занятиям, материалы текущего контроля и текущей аттестации, вопросы для подготовки к зачёту, задания и вопросы для самоконтроля, список рекомендуемой литературы и др.

Примерный вариант задания к зачету

Для заданной преподавателем предметной области (например, библиотека), выполните разработку клиент-серверного приложения в версии MVP (минимально жизнеспособный продукт):

1. Для заданной предметной области, разработайте список пользовательских требований и Техническое задание.
2. На основе методологии Scrum и Технического задания определите Product backlog и план спринтов для проектирования и разработки клиент-серверного приложения, создав Scrum board на подходящем для этого ресурсе.
3. Выполните проектирование схемы базы данных, и проектирование серверной части и клиентской части приложения на основе многослойной или микросервисной архитектуры.
4. Выполните программную разработку серверной части и клиентской части приложения, размещая исходный код проекта на ресурсе, поддерживающем систему контроля версий, например, https://github.com.

Примерный перечень вопросов к зачету

1. Методологии разработки ПО. Scrum, Kanban, XP, Microsoft Solutions Framework, Rational Unified Process и др.
2. Модели жизненного цикла ПО (Каскадная модель, Модифицированная каскадная модель, V-образная модель, Спиральная модель, Итеративная модель, Модель оценки уровня зрелости (CMMI) и др.
3. Гибкие методологии разработки ПО (Ajile). Управление проектом по Agile методологии Scrum.
4. Проектирование проекта с помощью UML. Диаграммы UML: вариантов использования, последовательности, классов, пакетов, деятельности, состояний развёртывания.
5. Архитектуры программного обеспечения: Многослойная архитектура (Layered Architecture), Многоуровневая архитектура (Tiered Architecture), Сервис-ориентированная архитектура (Service Oriented Architecture – SOA), Микросервисная архитектура (Microservice Architecture).
6. Принципы объектно-ориентированного проектирования SOLID. Принципы DRY, WET, KISS, YAGNI.
7. Протокол прикладного уровня HTTP. Его эволюция. Структура HTTP-запроса и ответа. URI, URL, URN.
8. Клиент-серверное взаимодействие.
9. Restful приложения. Архитектура REST. Принципы REST. Формат обмена данными JSON.
10. Протокол обмена структурированными сообщениями SOAP (Simple Object Access Protocol). Сравнение подходов SOAP и REST.
11. Инструменты создания качественного программного кода. Модульное тестирование. Журналирование событий.
12. Способы аутентификации в веб-приложениях. Аутентификация и авторизация. Куки и токены. Виды токенов: OAuth, JWT (JSON Web Token), SWT (Simple Web Token), SAML (Security Assertion Markup Language). Хеширование пароля.
13. Управление данными. Виды СУБД. Типы данных. Принципы управления данными.
14. История развития СУБД.
15. Основные виды СУБД.
16. Безопасность данных: политики безопасности данных, шифрование данных, управление доступом.
17. Управление доступом на основе ролей (RBAC) и атрибутов (ABAC).
18. Современные тенденции в управлении данными.
19. Основы CI/CD. Настройка контейнеризации через Docker.
20. Сравнение виртуализации и контейнеризации.
21. Три основных типа виртуализации.
22. Масштабирование приложений. Определения понятия «Распределенные системы». Характеристики распределенной системы.
23. Имена, идентификаторы и адреса.

24. Методы присваивания имен: централизованная система именования, децентрализованная система именования, иерархические системы именования.

25. Система доменных имен (DNS). Понятие хостинга, домена.

26. Безопасность веб-приложений. Основные угрозы и риски. Примеры атак и их последствий.

27. Методы обеспечения конфиденциальности. Методы защиты веб-приложений.

28. Использование кэширования для оптимальной производительности API и веб-приложений.

29. Типы механизмов кэширования, обычно используемых в API.

30. Документирование проекта: Устав проекта, Техническое задание, План управления рисками, План управления коммуникациями, Руководство пользователя.

ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ УО

Название учебной дисциплины, с которой требуется согласование	Название кафедры	Предложения об изменениях в содержании учебной программы учреждения высшего образования по учебной дисциплине	Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола)
Учебная дисциплина не требует согласования			

Заведующий кафедрой
многопроцессорных систем и сетей _____ И.Е.Андрушкевич
к.ф-м.н., доцент (подпись)

02.06.2025

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ К УЧЕБНОЙ ПРОГРАММЕ УО

на _____/_____ учебный год

№ п/п	Дополнения и изменения	Основание

Учебная программа пересмотрена и одобрена на заседании кафедры
_____ (протокол № ____ от _____ 202_ г.)

Заведующий кафедрой
_____ _____

УТВЕРЖДАЮ
Декан факультета
_____ _____