

ТЕРАКОР • ПРОФЕССИЯ В IT

современным IT-профессиям
для вашего успеха



Онлайн-формат



Практика
и проекты



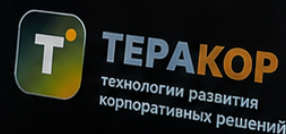
Поддержка
на всех этапах



ТЕРАКОР
технологии развития
корпоративных решений

СДЕЛАЙ ШАГ
К НОВОЙ
ПРОФЕССИИ

ЗАПИШИСЬ
СЕЙЧАС! →



BACKEND-РАЗРАБОТКА • 5 МЕСЯЦЕВ

Java-разработчик

От Java Core до Spring Boot, PostgreSQL, Kafka и микросервисов
на современном стеке и в формате проектной разработки.

Главное отличие ТЕРАКОР

Мы сами занимаемся современной коммерческой разработкой, поэтому обучение строится не вокруг устаревшей теории, а вокруг актуального стека, архитектуры, инженерных практик и проектов, максимально приближенных к реальной работе.

450 ак. ч

полная программа

150 ак. ч

живых занятий

90 000 ₽

полная стоимость

Сергей Валерьевич Герасименко

Основатель ТЕРАКОР • практикующий full-stack разработчик

18 000 ₽ / месяц × 5

при равномерной оплате полной программы

terakor.ru

Лицензированный учебный центр • онлайн-обучение • практика и проекты

1. О профессии

Профессия Java-разработчик — это цельная траектория от основ языка и ООП до разработки production-like backend-систем с PostgreSQL, Spring Boot, Apache Kafka, микросервисами, Docker и современными AI-интеграциями.

Главная идея программы

Не изучать технологии изолированно, а научиться собирать из них работающую систему: проектировать API, хранить данные, обеспечивать безопасность, обмениваться событиями, тестировать, контейнеризировать и поддерживать приложение.

- ✓ обучение ведется в живом формате с преподавателем;
- ✓ каждый большой этап завершается практической работой или проектом;
- ✓ стек ориентирован на современную Java backend-разработку;
- ✓ темы рассматриваются не только на уровне API, но и с объяснением архитектурных решений и типичных ошибок;
- ✓ при наличии подготовки студент может начать обучение с более поздней точки входа после тестирования или технического собеседования.

2. Результаты обучения

- ✓ Разрабатывать backend-приложения на Java 21 и понимать ключевые возможности современного Java API.
- ✓ Проектировать предметную модель, слои приложения и REST API.
- ✓ Работать с PostgreSQL: SQL, транзакции, индексы, планы выполнения и оптимизация запросов.
- ✓ Использовать JDBC, JPA/Hibernate и Spring Data в зависимости от задачи.
- ✓ Создавать приложения на Spring Boot, настраивать конфигурацию, валидацию, обработку ошибок и безопасность.
- ✓ Писать unit-, integration- и API-тесты.
- ✓ Интегрировать Apache Kafka: Producer/Consumer, consumer groups, offsets, Admin API, idempotence, transactions, Streams, Schema Registry и Connect.
- ✓ Проектировать событийное взаимодействие между сервисами и понимать delivery semantics.
- ✓ Разделять систему на микросервисы, контейнеризировать ее и запускать в Docker Compose.
- ✓ Использовать логирование, метрики и базовые практики observability.
- ✓ Понимать, где уместен Spring AI и как подключать LLM/AI-сервисы к Java-приложениям.
- ✓ Читать существующий код, находить проблемы, выполнять live coding и аргументировать инженерные решения на техническом собеседовании.

3. Нагрузка и формат

Этап	Живые занятия	Самостоятельная работа / проекты
Этап 1. Java Core	36 ак. ч.	72 ак. ч. практика и проект
Этап 2. Java Advanced	24 ак. ч.	48 ак. ч. практика
Этап 3. PostgreSQL и Data Access	20 ак. ч.	40 ак. ч. практика
Этап 4. Spring Boot Backend	34 ак. ч.	68 ак. ч. практика и проект

Этап 5. Apache Kafka	20 ак. ч.	40 ак. ч. лабораторные и проект
Этап 6. Microservices, Docker, Spring AI и собеседование	16 ак. ч.	32 ак. ч. итоговый проект и подготовка
ИТОГО	150 ак. ч.	300 ак. ч.

Общий объем программы: 450 академических часов. Распределение самостоятельной нагрузки может корректироваться под темп группы и выбранную точку входа.

4. Подробная программа

Модуль 1. Java: основы языка и среда разработки

16 ак. ч.

Результат модуля: Студент уверенно пишет консольные Java-приложения и понимает базовую модель выполнения программы.

- JDK 21, JVM, JRE, структура Java-проекта.
- Типы данных, операторы, условия, циклы, массивы.
- Методы, параметры, возвращаемые значения, область видимости.
- Работа со строками, датами и базовыми классами стандартной библиотеки.
- Отладка в IDE, чтение stack trace, базовая работа с Git.
- Практика: набор небольших задач + мини-проект.

Модуль 2. ООП, коллекции, исключения и generics

20 ак. ч.

Результат модуля: Студент проектирует объектную модель и применяет коллекции, интерфейсы, generics и исключения осознанно.

- Классы, объекты, инкапсуляция, наследование, полиморфизм.
- Абстрактные классы, интерфейсы, композиция.
- equals/hashCode/toString, immutable-объекты, record.
- Collections Framework: List, Set, Map, Queue.
- Comparable/Comparator, сортировки и выбор структуры данных.
- Generics: классы, методы, wildcards.
- Исключения: checked/unchecked, собственные исключения, обработка ошибок.
- Практика: предметная модель + сервисный слой без Spring.

Модуль 3. Java Advanced: Stream API, I/O, многопоточность

24 ак. ч.

Результат модуля: Студент владеет современными средствами Java и понимает основы конкурентного программирования.

- Lambda expressions, functional interfaces, method references.
- Stream API: map/filter/reduce/collect, grouping, partitioning.
- Optional и работа с null.
- Файлы, NIO.2, Path/Files, сериализация данных в JSON.
- Threads, Runnable/Callable, ExecutorService.
- Future/CompletableFuture, synchronized, locks, concurrent collections.
- Проблемы race condition, deadlock, visibility; основы Java Memory Model.
- Модульность JPMS и устройство современного Java-приложения.
- Практика: параллельная обработка данных и рефакторинг проекта.

Модуль 4. SQL и PostgreSQL**12 ак. ч.**

Результат модуля: Студент проектирует схему данных, пишет нетривиальные запросы и понимает основы производительности PostgreSQL.

- Реляционная модель, таблицы, ключи, ограничения, нормализация.
- SELECT, JOIN, подзапросы, CTE, агрегаты.
- INSERT/UPDATE/DELETE, транзакции и уровни изоляции.
- Индексы: B-tree, назначение и ограничения.
- EXPLAIN/EXPLAIN ANALYZE, чтение плана выполнения.
- JSON/JSONB, оконные функции, полнотекстовый поиск - прикладной обзор.
- Практика: схема данных для backend-сервиса + оптимизация запросов.

Модуль 5. JDBC, Data Access и миграции**8 ак. ч.**

Результат модуля: Студент понимает низкоуровневую работу Java с БД и готов к осознанному использованию ORM.

- JDBC API: Connection, Statement, PreparedStatement, ResultSet.
- Пулы соединений и HikariCP.
- DAO/Repository pattern без Spring Data.
- Транзакции на уровне приложения.
- Liquibase/Flyway: управление схемой и миграциями.
- Практика: подключение Java-приложения к PostgreSQL.

Модуль 6. Spring Core и Spring Boot**14 ак. ч.**

Результат модуля: Студент создает Spring Boot-приложение и понимает DI/IOC, конфигурацию и жизненный цикл компонентов.

- IoC/DI: зачем нужен контейнер и как он работает.
- Beans, scopes, component scanning, Java Configuration.
- Spring Boot 4: auto-configuration, starters, configuration properties.
- Profiles, внешняя конфигурация, environments.
- Bean lifecycle, validation, application events.
- Практика: перевод проекта на Spring Boot.

Модуль 7. REST API, JPA/Hibernate, Security и тестирование**20 ак. ч.**

Результат модуля: Студент создает защищенный REST backend с PostgreSQL и покрывает критическую логику тестами.

- REST: ресурсы, HTTP-методы, коды ответа, идиоматичность.
- Spring MVC / REST controllers, DTO, validation.
- Глобальная обработка ошибок и единый формат API error response.
- JPA/Hibernate: entities, persistence context, fetch strategies, N+1.
- Связи one-to-one, one-to-many, many-to-many.
- Spring Data JPA, query methods, JPQL/native queries.
- Транзакции в Spring, границы транзакций и типичные ошибки.
- Spring Security: authentication, authorization, password hashing, JWT / token-based подход.
- Unit testing: JUnit 5, Mockito.
- Integration testing: Spring Boot Test, Testcontainers.

- Практика: полноценный REST API с БД, Security и тестами.

5. Apache Kafka - отдельный углубленный блок, 20 ак. ч.

Kafka в программе выделена в самостоятельный инженерный модуль.

Цель - не только научиться отправлять и читать сообщения, а понимать устройство Kafka, гарантии доставки, consumer groups, транзакции, схемы сообщений, потоковую обработку, интеграцию с внешними системами и эксплуатационные аспекты.

Почему блок глубже типового курса

Помимо базовых Producer/Consumer/Admin API включены архитектура Kafka 4 и KRaft, стратегии партиционирования и rebalance, ручное управление offsets, idempotence и transactions, Schema Registry, Kafka Streams, Connect, observability, performance tuning, security и интеграция со Spring Boot. Практика строится вокруг событийного микросервисного сценария.

№	Тема	Часы
1	Архитектура Kafka 4, KRaft, topics, partitions, replication, ISR, durability	3
2	Producer API: batching, acks, retries, idempotence, partitioning, serialization	2
3	Consumer API: consumer groups, rebalance, offsets, commits, delivery semantics	3
4	Admin API и эксплуатационные операции	2
5	Transactions и exactly-once: transactional producer, read_committed, EOS	2
6	Schema Registry и эволюция контрактов: Avro / Protobuf / JSON Schema	2
7	Kafka Streams и stateful processing: KStream/KTable, joins, state stores	2
8	Connect, monitoring, performance, security, Spring Kafka + итоговая практика	4

5.1. Архитектура и базовая модель Kafka - 3 ак. ч.

- Event-driven architecture: события, команды и сообщения; когда Kafka уместна, а когда лучше другой брокер.
- Kafka 4 и KRaft: роль controller quorum, brokers, metadata, отказ от ZooKeeper.
- Topics, partitions, segment files, offsets; порядок сообщений и его границы.
- Replication factor, leader/followers, ISR, min.insync.replicas.
- Retention по времени/размеру, log compaction.
- Практика: запуск multi-broker Kafka в Docker Compose, создание топиков и анализ распределения партиций.

5.2. Producer API - 2 ак. ч.

- KafkaProducer и ProducerRecord; key/value, headers, timestamp.
- Serializer: String/JSON и пользовательские сериализаторы.
- Partitioner, ключи и сохранение порядка.
- acks, retries, delivery.timeout.ms, linger.ms, batch.size, compression.

- Idempotent producer и защита от дубликатов при retry.
- Практика: Java Producer с callback, headers и осмысленным ключом партиционирования.

5.3. Consumer API и offsets - 3 ак. ч.

- KafkaConsumer, subscribe/assign, poll loop.
- Consumer groups и масштабирование чтения.
- Partition assignment strategies и cooperative rebalance.
- Rebalance listeners и корректное завершение consumer.
- auto commit vs manual commit; commitSync/commitAsync.
- at-most-once / at-least-once / exactly-once: где возникает дубль или потеря.
- Практика: несколько consumer-инстансов, rebalance, ручное подтверждение обработки и replay сообщений.

5.4. Admin API - 2 ак. ч.

- AdminClient и асинхронная модель KafkaFuture.
- Создание/изменение/удаление topics.
- Описание partitions, configs, consumer groups и offsets.
- Увеличение числа partitions и последствия для key-based ordering.
- Удаление/перенос offsets для тестовых и эксплуатационных сценариев.
- Практика: AdminApp для аудита топиков и consumer groups.

5.5. Transactions и exactly-once - 2 ак. ч.

- Идемпотентность и транзакции - разные задачи.
- transactional.id, initTransactions, begin/commit/abort.
- sendOffsetsToTransaction и consume-transform-produce.
- isolation.level=read_committed.
- Exactly-once semantics и реальные ограничения EOS.
- Практика: транзакционный pipeline без публикации частично обработанных данных.

5.6. Schema Registry и контракты сообщений - 2 ак. ч.

- Почему JSON без контракта плохо масштабируется между командами.
- Avro, Protobuf, JSON Schema: сравнение подходов.
- Schema Registry и subject naming strategies.
- Backward/forward/full compatibility и эволюция схем.
- Poison pill, несовместимые сообщения и стратегии обработки ошибок.
- Практика: версионирование события без поломки consumer.

5.7. Kafka Streams - 2 ак. ч.

- KStream и KTable, topology и processor model.
- Stateless transformations: filter/map/flatMap/branch.
- State stores и changelog topics.
- Joins, aggregations, windowing.
- Processor API - когда DSL недостаточно.
- Практика: потоковая агрегация заказов и материализованное состояние.

5.8. Connect, эксплуатация и Spring Kafka - 4 ак. ч.

- Kafka Connect: source/sink connectors, workers, tasks, converters.
- Интеграция PostgreSQL/Kafka на концептуальном уровне, CDC как развитие темы.

- Метрики broker/producer/consumer: lag, throughput, request latency, under-replicated partitions.
- Performance tuning: размер batch, compression, fetch settings, partitions и компромиссы throughput/latency.
- Security: TLS, SASL, ACL - архитектурный обзор и минимальная конфигурация.
- Spring for Apache Kafka: KafkaTemplate, @KafkaListener, listener containers, error handlers.
- Retry, backoff и Dead Letter Topic; идемпотентная бизнес-обработка.
- Практика: Spring Boot producer/consumer + DLT + metrics; финальная интеграция в микросервисный проект.

Практический результат блока Kafka

- ✓ Поднять Kafka 4 в Docker Compose и диагностировать состояние кластера.
- ✓ Спроектировать топик, ключи сообщений и количество партиций под бизнес-сценарий.
- ✓ Реализовать Producer и Consumer на Java/Spring Boot.
- ✓ Осознанно выбрать стратегию commit и гарантии доставки.
- ✓ Реализовать retry + Dead Letter Topic и защиту от повторной бизнес-обработки.
- ✓ Использовать транзакции Kafka в consume-transform-produce сценарии.
- ✓ Версионировать контракт события через Schema Registry.
- ✓ Реализовать простую stateful-обработку в Kafka Streams.
- ✓ Собрать базовые метрики и объяснить причины consumer lag.

Модуль 9. Микросервисы, Docker и observability**10 ак. ч.**

Результат модуля: Студент понимает ключевые компромиссы микросервисной архитектуры и собирает несколько сервисов в единую систему.

- Monolith vs modular monolith vs microservices: критерии выбора.
- Разделение bounded contexts и контрактов между сервисами.
- REST vs event-driven integration.
- Ошибки распределенных систем: timeout, retry, duplicate, partial failure.
- Паттерны Saga, Outbox, idempotency - практический обзор.
- Dockerfile, image, container, network, volumes.
- Docker Compose для PostgreSQL, Kafka и нескольких сервисов.
- Логирование, correlation id, health checks, метрики.
- Практика: разбиение проекта на сервисы и запуск инфраструктуры одной командой.

Модуль 10. Spring AI и итоговая архитектура проекта**2 ак. ч.**

Результат модуля: Студент понимает, где AI уместен в Java backend, и умеет встроить базовый AI-сценарий в Spring Boot-приложение.

- Spring AI: ChatClient, prompts и structured output - прикладной обзор.
- RAG как архитектурная идея: embeddings, vector stores и retrieval - обзор архитектуры.
- Ограничения LLM-интеграций: data leakage, hallucinations, validation и контроль результата.
- Практика: подключение одной AI-функции к существующему Spring Boot backend.
- Архитектурный review итогового проекта перед финальной защитой.

Модуль 11. Подготовка к техническому собеседованию**4 ак. ч.**

Результат модуля: Студент систематизирует знания, тренируется отвечать на вопросы по Java Backend-стеку, проходит live coding и пробное техническое собеседование с обратной связью.

- 1 ак. ч. - Java Core, ООП, коллекции, generics, исключения, Stream API, многопоточность и JVM: типовые вопросы и разбор ответов.
- 1 ак. ч. - SQL/PostgreSQL, Spring Boot, JPA/Hibernate, Security, Kafka и микросервисы: вопросы по архитектуре и практическим сценариям.
- 1 ак. ч. - Live coding и чтение чужого кода: небольшая задача, поиск ошибок, рефакторинг и объяснение решения вслух.
- 1 ак. ч. - Пробное техническое собеседование по проекту Order Platform с разбором сильных и слабых мест и персональными рекомендациями.

6. Проекты в портфолио

Проект 1. Java Domain Application

Стек: Java Core, ООП, коллекции, исключения, generics.

Результат: Цель: спроектировать предметную модель и сервисный слой без фреймворков.

Проект 2. Data Service

Стек: Java, JDBC, PostgreSQL, миграции.

Результат: Цель: приложение с реляционной моделью, транзакциями и оптимизированными SQL-запросами.

Проект 3. Spring Boot REST API

Стек: Spring Boot, REST, JPA, PostgreSQL, Security, Testing.

Результат: Цель: защищенный backend с полноценным API и integration-тестами.

Итоговый проект. Order Platform

Стек: Java 21, Spring Boot, PostgreSQL, Kafka 4, Microservices, Docker, Security, Testing, Observability, Spring AI.

Результат: Цель: production-like система с синхронным и событийным взаимодействием сервисов.

7. Итоговый проект Order Platform - рекомендуемая архитектура

- Order Service - создание и изменение заказов, PostgreSQL, REST API.
- Payment Service - обработка платежного сценария и публикация событий.
- Notification Service - подписка на Kafka events и уведомления.
- Analytics/Stream Service - агрегация событий через Kafka Streams.
- Kafka topics с продуманными ключами, retry/DLT и контрактами сообщений.
- Schema Registry для эволюции событий.
- Docker Compose: сервисы + PostgreSQL + Kafka + вспомогательная инфраструктура.
- Spring Security, unit/integration tests, Testcontainers.
- Логирование, correlation id, health endpoints, базовые метрики.
- AI-функция (опционально): генерация краткого резюме заказа/обращения или интеллектуальный помощник оператора через Spring AI.

8. Контроль знаний и защита

- ✓ Практические задания после ключевых тем.
- ✓ Code review решений преподавателем.
- ✓ Промежуточные мини-проекты.
- ✓ Отдельный блок 4 ак. ч. подготовки к техническому собеседованию и пробное интервью.
- ✓ Итоговая защита Order Platform: архитектура, код, API, данные, Kafka, тестирование и аргументация решений.

9. Для кого подходит программа

- ✓ Новичкам, которые хотят пройти цельный путь до Java Backend-разработки.
- ✓ Разработчикам с Java Core, которым нужны Spring Boot, PostgreSQL, Kafka и микросервисы.
- ✓ Разработчикам, которые хотят систематизировать знания и перейти к более сложным backend-задачам.
- ✓ Специалистам, готовящимся к техническому собеседованию по Java backend стеку.

Гибкая точка входа

Если часть программы уже освоена, студент может пройти Java-тест или техническое собеседование и начать с подходящего этапа. Это позволяет не повторять уже знакомый материал и уменьшить стоимость обучения.

10. Формат обучения

Живые занятия

Преподаватель объясняет материал, пишет код, разбирает вопросы и показывает логику принятия решений.

Практика с первых занятий

Каждая технология закрепляется кодом и задачами, а не только конспектом.

Code Review

Разбор качества кода, структуры, ошибок и альтернативных решений.

Проектное обучение

Технологии связываются в несколько законченных приложений и единый итоговый проект.

Подготовка к собеседованию

В конце программы выделено 4 ак. ч. на подготовку к техническому собеседованию: вопросы по Java Backend-стеку, live coding, обсуждение архитектуры Order Platform и пробное интервью с обратной связью.

Актуализация программы

Стек и практики обновляются с учетом современной коммерческой разработки и собственных проектов ТЕРАКОР Digital.

11. Преподаватель

Сергей Валерьевич Герасименко

Практикующий full-stack разработчик, технический директор и основатель ТЕРАКОР и ТЕРАКОР Digital. Основные программы центра строятся на сочетании преподавательского опыта и собственной практики разработки программных систем.

Почему это важно

Студент учится не только пользоваться API библиотек, но и понимать, зачем выбирается конкретный подход, где он ломается, какие компромиссы возникают в архитектуре и как технология ведет себя в составе реальной системы.

12. Стоимость

Полная программа

90 000 ₽

Ориентир по месяцам

18 000 ₽ × 5

- ✓ 450 ак. ч. общей нагрузки
- ✓ 150 ак. ч. живых занятий
- ✓ 4 ак. ч. подготовки к техническому собеседованию
- ✓ Практические задания и проекты
- ✓ Code Review
- ✓ Итоговый проект
- ✓ Материалы курса и рекомендации по развитию

При наличии подтвержденной подготовки возможна более поздняя точка входа и меньшая стоимость. Актуальные условия проверяйте на странице профессии.

13. Следующий шаг

Начните с определения своего уровня Java

Пройдите бесплатный тест по Java или запишитесь на консультацию, чтобы подобрать оптимальную точку входа в программу.

[Открыть сайт ТЕРАКОР](#) • [Профессия Java-разработчик](#)