УДК 004.9

Мясоедов А.А.
студент
Воронежский государственный лесотехнический университет
имени Г. Ф. Морозова
Россия, Воронеж

РЕАЛИЗАЦИЯ БИБЛИОТЕКИ УПРАВЛЕНИЯ МАШИНОЙ СОСТОЯНИЙ В ПАМЯТИ ПРИЛОЖЕНИЯ

Аннотация: Статья посвящена проблемам реализации машины состояний для сущностей в информационных системах. Рассмотрена реализация библиотеки для управления машиной состояний с помощью языка программирования Kotlin. В публикации изложены подходы к реализации машин с конечным числом состояний, хранящихся в памяти приложения, для управления жизненным циклом сущностей информационной системы.
Ключевые слова: машина состояний, конечный автомат, событие, JVM, Kotlin.

Myasoedov A.A.
student
Voronezh State Forest Engineering University named after G. F. Morozov
Russia, Voronezh

IMPLEMENTATION OF A STATE MACHINE MANAGEMENT LIBRARY IN APPLICATION MEMORY

Abstract: The article is devoted to the problems of implementing a state machine for entities in information systems. The implementation of a library for managing a state machine using the Kotlin programming language is considered. The publication presents approaches to implementing machines with a finite number of states stored in the application memory for managing the life cycle of entities in an information system. *Keywords: state machine, finite state machine, event, JVM, Kotlin.*

В современной разработке программного обеспечения и построения информационных систем управление состоянием данных является одной из ключевых задач. Особенно это актуально в системах, работающих с документами, где необходимо отслеживать их жизненный цикл: создание, редактирование, утверждение, публикацию и другие события. Машина состояний — это математическая модель, которая позволяет формализовать переходы между состояниями объекта в зависимости от событий. В эту формализацию входят: правила перехода, условия, при которых может быть осуществлена смена состояния, побочные и сторонние эффекты при изменении состояния и другие подобные события.

Целью данной работы является создание библиотеки для управления машиной состояния [1] различных сущностей. Для реализации был выбран язык программирования Kotlin, так как он имеет глубокую поддержку функционального и контекстно-ориентированного программирования и отличную совместимость с JVM экосистемой.

Перед разработкой был проведён анализ существующих решений:

- Spring Statemachine [2]. Spring Statemachine является мощной библиотекой, встроенной в экосистему Spring. Она позволяет описывать машину состояний декларативно и имеет широкую функциональность, а также возможность гибкой модификации. Однако данное решение недостатки, а именно: избыточность для простых сценариев, слишком тесная связь со Spring экосистемой;
- Tinder. Tinder это легковесная библиотека для создания машин состояний на Kotlin [3]. Она предоставляет функциональность для описания различных состояний, переходов между ними, сторонних эффектов при переходах. Из минусов данной библиотеки можно выделить отсутствие возможности

сохранения состояния в персистентное хранилище и трудность модификации;

– Самописные библиотеки. Данные решения часто создаются для решения конкретной задачи и не являются универсальными.

Было решено реализовать легковесную библиотеку машины состояний на Kotlin, которая соответствует следующим требованиям:

– Декларативное описание состояний и переходов;
– Интеграция с корутинами для асинхронных операций;
– Возможность гибкой модификации библиотеки.

На рисунке 1 представлена диаграмма компонентов использования библиотеки.

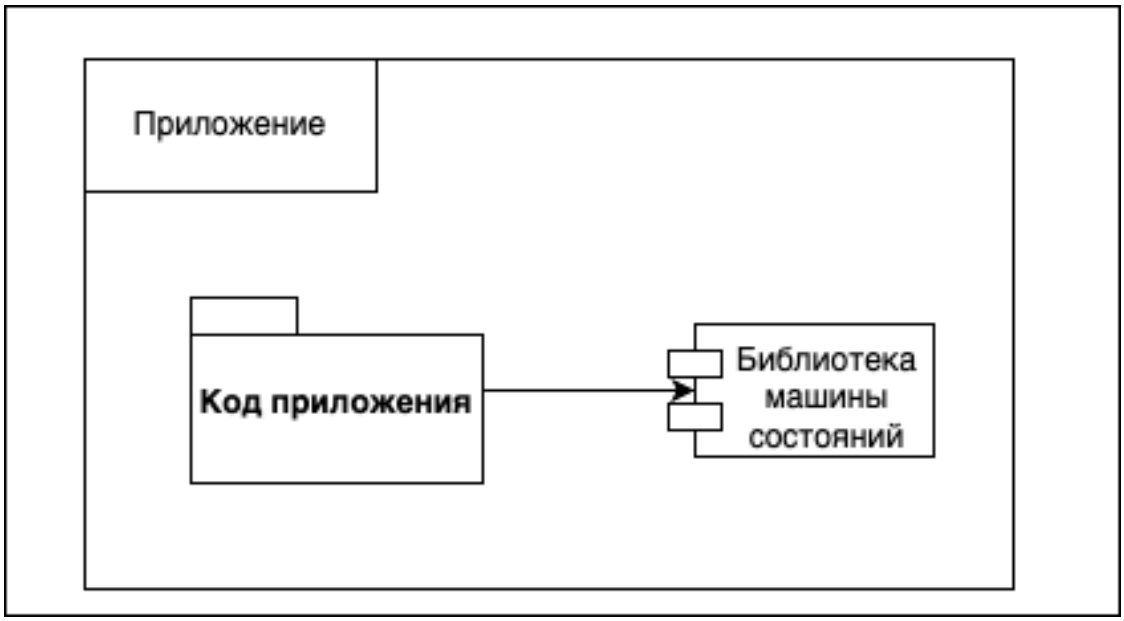

Рис. 1 – Диаграмма компонентов использования библиотеки машины состояний.

Основной класс библиотеки StateMachine <STATE, ACTION> отвечает за совершение переходов, проверку их корректности и хранение состояний отдельных сущностей. Action и State пользовательские перечисления, которые зависят от конкретной бизнес области. На рисунке 2 представлена диаграмма классов машины состояний.

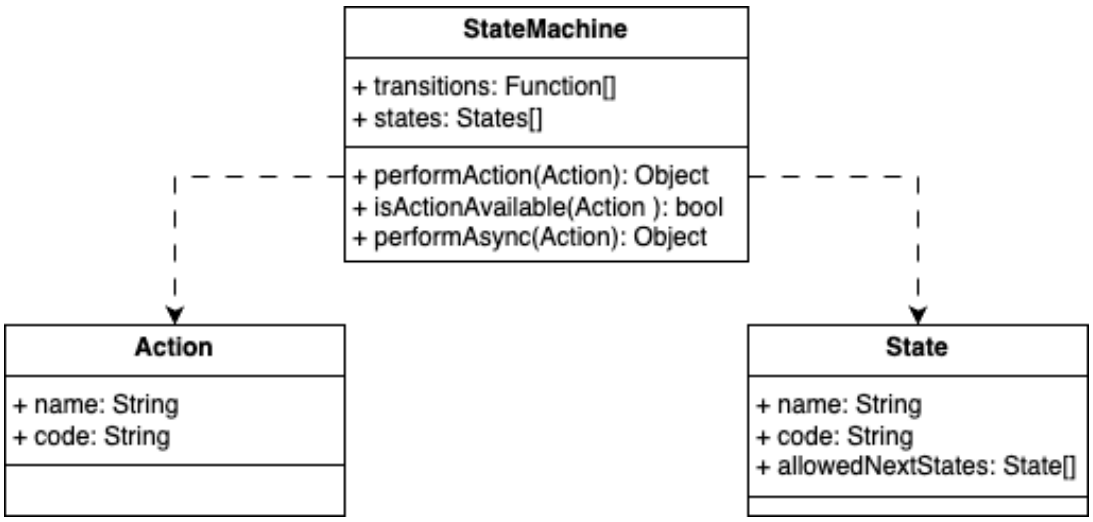

Рис. 2 – Диаграмма классов машины состояний.

В классе StateMachine поле transitions отвечает за переходы между состояниями, а поле states за список возможных состояний. Метод isActionAvailable проверяет допустимость перехода. Методы performAction и performAsync совершают действия, связанные с переходами синхронно и асинхронно с помощью корутин соответственно.

Разработанная библиотека может быть использована в различных системах, требующих управления жизненным циклом документов или других сущностей с изменяемым состоянием. В будущем возможно расширение функционала с помощью добавления персистентности хранения состояний во внешнем источнике.

Использованные источники:

1. Implementing simple state machines with Java enums [Электронный ресурс]// www.baeldung.com - URL: https://www.baeldung.com/java-enum-simple-state-machine (дата обращения: 10.05.2025)

2. Spring Statemachine [Электронный ресурс]// docs.spring.io - URL: https://docs.spring.io/spring-statemachine/docs/4.0.0/reference/index.html (дата обращения: 10.05.2025)

3. Kotlin DSL [Электронный ресурс]// www.jetbrains.com - URL: https://www.jetbrains.com/help/teamcity/kotlin-dsl.html (дата обращения: 10.05.2025)