Tool Depot Rental Checkout Microservice

Software Design Document

Author: Troy Landers

Introduction

This document presents the design of Tool Depot's Rental Checkout Microservice (RCM). This service is designed to fulfill the functional requirements outlined in the Tool Depot Rental Checkout Functional Specification. The RCM facilitates tool rentals for Tool Depot's clerks and customers by managing the checkout process and tool inventory retrieval.

The primary audiences of this document are the engineering, testing, and the deployment teams.

The document focuses exclusively on the RCM's design providing a roadmap for implementation while maintaining flexibility for future enhancements.

Microservice Architecture

The RCM will be built using a tiered, microservice architecture to ensure scalability, modularity, and maintainability. The tiers consist of the three layers shown in Diagram 1 - the REST API layer, the Services layer, and the Repository layer. Each layer should be implemented such that it is self-contained with minimal dependencies on the other layers. Diagram 1 also shows a database layer which is outside the scope of this document.

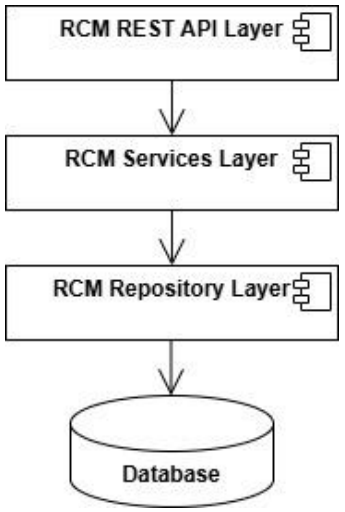

Diagram 1 - Microservice Architecture

RCM REST API Layer

The REST API Layer serves as the primary interface for external clients, providing REST endpoints to manage the checkout process and tool inventory. It hides the details of the business logic required for checkout. It abstracts the complexities of business logic, ensuring a seamless interaction with the system while enforcing input validation and error handling.

RCM Services Layer

The Services Layer exposes a Spring Boot service interface that is utilized by the REST API Layer to realize its endpoints. The Services Layer encapsulates the business logic involved in completing a rental transaction and hides the details of the layers beneath it.

RCM Repository Layer

The Repository Layer exposes a Spring Boot service interface that is utilized by the Services Layer. The Repository Layer hides the details of the underlying database technology.

Interface Design

The REST API Layer exposes three REST endpoints - Checkout, Find All Tools, and Get Tool - that can be utilized by external clients and front end components. The endpoints define the contract with clients, ensuring consistency and reliability.

All endpoints return a resultCode and message to communicate error information. These fields are unused and should be ignored when the endpoint returns a successful HTTP response.

Checkout Endpoint

The Checkout endpoint executes the checkout process and returns rental transaction data.

URL: /api/checkout
Method: POST
Request Body Inputs (JSON)

- toolCode : String - 4-character code of tool to be rented
- rentalDays : int - total number of days that the tool will be rented
- discountPercent : int - must be in range [0,100]
- checkoutDate : Date (MM/dd/yy)

Response JSON

- tool : object
    - toolCode : String
    - brand : String - brand associated with input toolCode

        - toolType : String - type of tool
    - rentalDays : int
    - checkoutOutDate : Date (MM/dd/yy)
    - dueDate : Date (MM/dd/yy) - date that tool must be returned by
    - dailyRentalCharge : decimal (#.##) - daily charge associated with tool type
    - chargeDays : int - days that are charged rent, will be in range [0, rentalDays]
    - preDiscountCharge : decimal (#.##) - full charge amount assuming no discount
    - discountPercent : int
    - discountAmount : decimal (#.##) - amount to be discounted from pre-discount charge
    - finalCharge : decimal (#.##) - amount to be charged after discount is applied
    - resultCode : int
        - 0 : successful checkout
        - 1 : invalid discountPercent is input
        - 2 : invalid rentalDays is input
        - 3 : invalid toolCode is input
    - message : String - ignore if no error, otherwise the error reason is given

HTTP Response Codes

- 201 : Created - checkout is successful
- 422 : Unprocessable Entity - see resultCode and message for error details
- 5xx : Unexpected server error

Find All Tools Endpoint

The Find All Tools endpoint returns all tools in the Tool Depot tool repository.

URL: /api/tools
Method: GET
Request Inputs - none
Response JSON

- tools : array of Tool objects
    - toolCode : String
    - toolType : String
    - brand : String
    - dailyCharge : decimal (#.##)
    - weekdayCharge : boolean
    - weekendCharge : boolean
    - holidayCharge : boolean
- resultCode : int
    - 0 : success
- message : String - ignore if no error, otherwise the error reason is given

HTTP Response Codes

- 200 : Success
- 5xx : Unexpected server error

Get Tool Endpoint

The Get Tool endpoint fetches the requested tool from the tool repository.

URL: /api/tools/{toolCode}
Method: GET
Request Querystring Inputs
- toolCode : String

Response JSON
- toolCode : String
- toolType : String
- brand : String
- dailyCharge : decimal (#.##)
- weekdayCharge : boolean
- weekendCharge : boolean
- holidayCharge : boolean
- resultCode : int
    - 0 : success
- message : String - ignore if no error, otherwise the error reason is given

HTTP Response Codes
- 200 : Success
- 404 : Tool not found
- 5xx : Unexpected server error

Component Design

The components of the RCM are the REST API Components, the Service Components, and the Repository Components.

REST API Components

The REST API Components consist of the RentalApiController and the ToolApiController shown in Diagram 2.

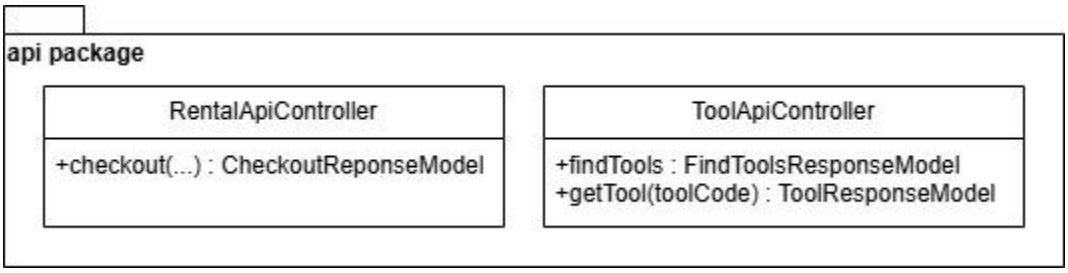

Diagram 2 - REST API Components

The RestApiController is a Spring Boot Rest Controller that exposes the Checkout endpoint. Its main responsibilities are to provide a REST endpoint for the CheckoutService so that external clients can access the service, and to return errors in a manner that clients can understand. There should be no checkout-related business logic in this component other than input validation and handling output data.

The ToolApiController is a Spring Boot Rest Controller that exposes the Find Tools and Get Tool Endpoints. Its main responsibilities are also to provide REST endpoints for these services, and to return errors in a manner that clients can understand. There should also be no business logic in these services aside from input validation and output handling.

Service Components

The Service Components shown in Diagram 3 consist of the CheckoutService, the ToolService, the PricingService, and the RentalPeriod Service. Each of these services are implemented as Spring Boot Service beans.

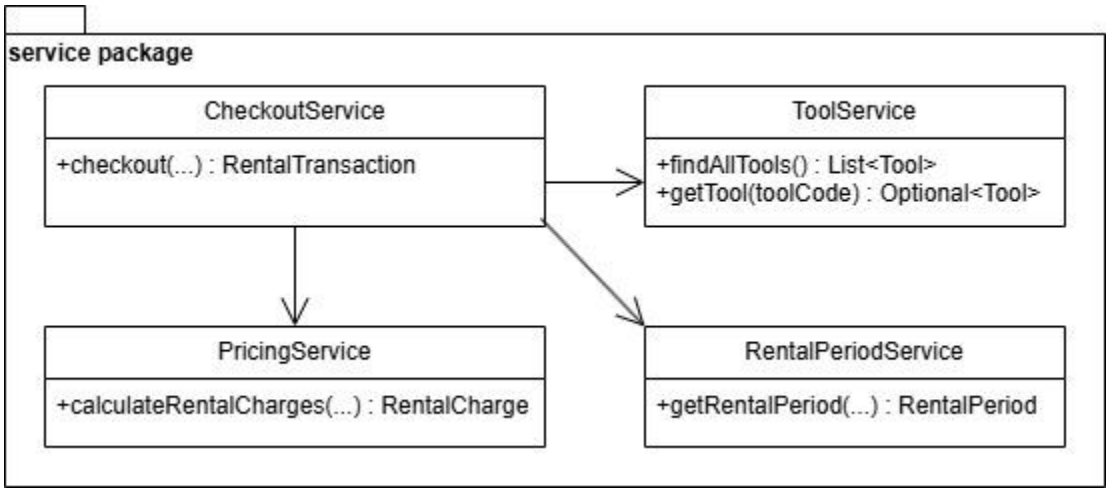

Diagram 3 - Service Components

The CheckoutService orchestrates the process for checking out a rental tool. It utilizes the other services and the RentalTransactionRepository to accomplish this task. Diagram 4 shows the checkout process and how each of the other components are involved.

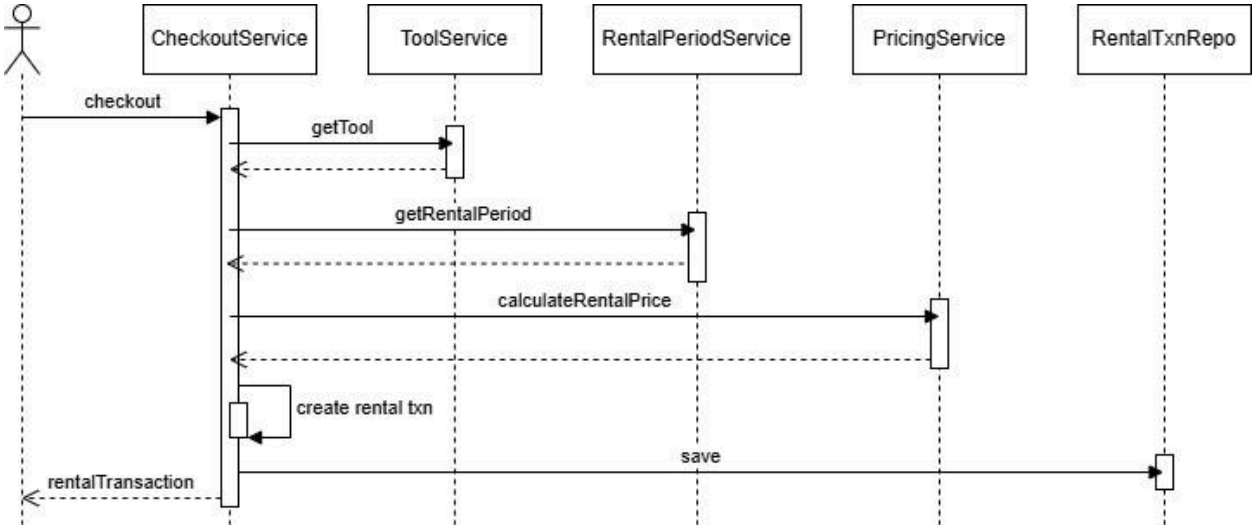

Diagram 4 - Checkout Sequence

The ToolService is responsible for managing the tools that are rented by Tool Depot. It exposes methods for retrieving all tools in inventory, and for retrieving information about a specific tool. The ToolService communicates with the ToolRepository to accomplish these tasks.

The RentalPeriodService determines the amount of days that are charged for a given tool and time period. This calculation can be complex and has the possibility of changing in the future so it is encapsulated in this service allowing it to change without impacting other parts of the RCM. The RentalPeriodService uses the HolidayRepo to get information about holidays that might impact the rental charges.

The PricingService accepts as input the tool type, the rental period, and any discounts, and uses them to determine the final price of the rental. Pricing can also be complex and can change as business needs evolve so all pricing calculations are encapsulated in this service.

Repository Components

The RCM's Repository Components shown in Diagram 5 are the ToolRepo, the HolidayRepo, and the RentalTransactionRepo. These repositories are used by the Service Components to retrieve and persist tools, holidays, and rental transactions to the database. These components are an abstraction layer above the database and isolate the RCM from changes that might occur with the underlying database technology.

Diagram 5 - Repository Components