****

**软件工程课程设计**

**软件需求规约文档**

**Easy Deploy：一站式AI训练服务平台**

项目仓库URL: https://github.com/Data-Science-Tech/UniDataTeam.git

|  |  |  |
| --- | --- | --- |
| **贾文超** |  | **2253968** |
| **秦成** |  | **2250397** |
| **张扬** |  | **2253722** |
| **周政宇** |  | **2252042** |

2025年1月4日

**目录**

[1. **项目简介** 1](#_Toc186889175)

[1.1 **项目背景** 1](#_Toc186889176)

[1.2 **项目目标** 1](#_Toc186889177)

[1.3 **项目范围** 2](#_Toc186889178)

[1.4 **利益相关者** 2](#_Toc186889179)

[2. **术语表** 4](#_Toc186889180)

[3. **项目约束** 4](#_Toc186889181)

[3.1 **时间约束** 4](#_Toc186889182)

[3.2 **资源约束** 5](#_Toc186889183)

[3.3 **质量约束** 5](#_Toc186889184)

[3.4 **用户约束** 5](#_Toc186889185)

[3.5 **解决方案约束** 5](#_Toc186889186)

[3.6 **外部系统约束** 6](#_Toc186889187)

[3.7 **预期工作环境约束** 6](#_Toc186889188)

[4. **功能性需求** 7](#_Toc186889189)

[4.1 **训练配置模块** 7](#_Toc186889190)

[4.1.1 **基于场景的建模** 7](#_Toc186889191)

[4.1.2 **基于类的建模** 9](#_Toc186889192)

[4.1.3 **基于行为的建模** 10](#_Toc186889193)

[4.2 **训练监控模块** 11](#_Toc186889194)

[4.2.1 **基于场景的建模** 11](#_Toc186889195)

[4.2.2 **基于类的建模** 12](#_Toc186889196)

[4.2.3 **基于行为的建模** 13](#_Toc186889197)

[4.3 **任务管理模块** 14](#_Toc186889198)

[4.3.1 **基于场景的建模** 14](#_Toc186889199)

[4.3.2 **基于类的建模** 16](#_Toc186889200)

[4.3.3 **基于行为的建模** 17](#_Toc186889201)

[5. **非功能性需求** 19](#_Toc186889202)

[5.1 **可用性需求** 19](#_Toc186889203)

[5.2 **兼容性需求** 19](#_Toc186889204)

[5.3 **可维护性需求** 19](#_Toc186889205)

[5.4 **可扩展性需求** 19](#_Toc186889206)

[6. **问题探讨** 20](#_Toc186889207)

[6.1 **为什么数据和算法管理部分的数据集转化和算法适配是小组完成，但要作为外部系统？** 20](#_Toc186889208)

[6.2 **为什么可以保证算法读取原始数据集和统一数据集的数据集是相同的？** 20](#_Toc186889209)

[7. **参考文献** 23](#_Toc186889210)

1. **项目简介**

1.1 **项目背景**

随着人工智能特别是大模型训练技术的迅速发展，数据处理和算法优化已成为行业关注的核心问题。然而，虽然大模型在多个领域（如自动驾驶、自然语言处理、图像识别等）有着广泛的应用前景，但对于对于初学者和非技术用户而言，当前大模型的学习和开发流程仍面临许多挑战，学习和实践的门槛依然较高：

**数据格式不统一。**不同的自动驾驶数据集（如KITTI、nuScenes等）采用不同的数据格式，因此算法也需要不同读取数据集的策略。对于初学者而言，他们需要对其他数据集进行复杂费事的转化，才能在已有算法和模型中进行验证验证。此类繁琐的工作增加了学习的复杂性，也降低了效率。

**算法适配难度高。**为了让算法能够兼容不同的数据集，开发者往往需要定制适配工具，这极大增加了算法开发和迁移的难度。这种高成本的定制化工作不仅浪费了时间和资源，还限制了算法的可扩展性。

**入门门槛高，环境配置复杂。**大模型的训练不仅需要强大的算力支持，还要求开发者具备一定的技术水平才能配置训练环境。然而，复杂的依赖安装、硬件配置和环境搭建对于初学者和非技术用户而言是巨大的障碍。这使得很多有潜在用户无法快速上手，并高效开展模型训练工作。

**全栈式服务不足**： 当前市面上的大多数平台只能提供单一功能，如算力租赁、算法开发工具或数据存储服务，或者提供上述功能的集成，如Hugging Face。无法满足用户从数据准备到模型训练的全流程需求，用户可以通过跨平台操作，这增加了学习成本，并导致效率低下。

1.2 **项目目标**

本项目旨在打造一个全面、简化的AI训练平台，满足用户从**数据集获取**到**模型训练**的全流程需求，即最终目标是通过同一领域数据集的统一，实现算法无需进行修改适配不同的数据集。我们将通过python脚本工具，消除不同数据集（如KITTI、nuScenes等）格式不一致的问题，帮助用户快速处理数据，避免数据预处理的瓶颈。平台将提供无代码的操作界面，使用户能够通过简单的配置完成从数据集获取到模型训练的整个过程，无需编写代码或进行复杂的环境配置。

此外，平台将集成训练配置、训练监控、任务管理等功能，提供一站式服务，极大提高用户效率。通过训练流程的可视化操作界面，降低技术门槛，使初学者和非技术用户也能轻松上手并顺利完成AI算法的开发和训练。

1.3 **项目范围**

本项目聚焦于自动驾驶领域的视觉感知算法，主要包括 **FAST R-CNN** 和 **SSD**两种常用的深度学习目标检测算法。同时，为了展示算法在不同数据集上的应用，本项目选择了 **KITTI** 和 **nuScenes** 数据集作为示例。由于不同数据集格式的多样性，本项目通过**Python脚本**，实现了数据集格式转换为统一格式的数据集，确保这些数据能够被算法一致处理和使用，从而减少了数据预处理的复杂性，提升了训练效率。

在功能方面，项目实现从数据集获取到模型训练的整个过程，提供了三个核心模块：**训练配置模块**、**训练监控模块**和**任务管理模块**。用户可以通过平台选择适合的算法（如FAST R-CNN或SSD）、合适的数据集（如KITTI 和 nuScenes）并配置训练参数（如学习率、批次大小等）。用户可以根据对应配置，启动训练任务。训练过程中，平台实时反馈训练进度，并在训练结束后展示训练效果的各项指标，如模型损失的变化曲线（折线图），帮助用户直观了解模型训练的情况。此外，平台还支持**任务管理**功能，用户可以查询历史训练记录、查看训练配置详情以及实时监控当前训练任务的状态。

最终，系统输出的主要成果为经过训练的模型和相关的训练日志信息，用户可以通过上述任务管理模块进行下载和保存。

通过这些功能的整合，本项目为用户提供了一个高效、便捷的模型训练平台，简化了数据处理、算法选择、训练监控等多个环节，使用户能够快速启动和管理训练任务，并最终输出的相应的训练模型和训练日志。

1.4 **利益相关者**

本部分将分析项目中的主要利益相关者，明确他们的需求，以确保系统能有效解决问题并提升可用性。通过对AI初学者、算法开发人员、数据采集工程师和软件工程师的需求分析，为系统需求获取提供了清晰的指导。



李同学作为AI初学者，尚处于学习阶段，缺乏深入的编程和数学背景。对他而言，系统的易用性和直观性是首要需求。李同学希望通过一个简单的界面快速完成AI训练的完整流程，以便更好地理解AI训练的操作与原理。因此，系统需要提供一个清晰、用户友好的界面，帮助他逐步掌握模型训练的核心概念，并能够通过自动化的流程减少配置难度，提升学习效率。



张先生是一名具有丰富经验的算法开发人员，擅长算法的优化。在他的工作中效率尤为重要。他希望系统能够支持快速完成训练配置，并且提供实时监控训练数据的功能，帮助他节省时间，专注于算法的优化与调试。因此，系统需要具备高效的训练配置工具、实时的训练数据监控功能，并支持算法的灵活调优，以便张先生能够快速发现问题并优化训练过程，提升算法性能。



李小姐是数据采集工程师，负责从多个渠道收集原始数据，并对其进行预处理。她的主要需求是，采集到的数据集能够直接在多种算法上运行，从而验证数据集的合理性。为此，系统需要支持多种常见的算法，帮助李小姐验证数据的完整性、有效性和适用性。



张先生作为软件工程师，主要关注系统的架构设计、代码可维护性和扩展性。在系统开发过程中，他需要确保系统具有清晰的模块化结构，能够快速理解并进行功能扩展。同时，良好的文档和注释也是他关注的重点，帮助团队成员更高效地进行协作开发。因此，系统必须采用模块化设计，确保每个模块的功能明确、独立性强，并提供详尽的API文档和开发指南，以确保系统的稳定性和可维护性，支持后期的持续优化和扩展。

2. **术语表**

|  |  |
| --- | --- |
| 术语 | 解释 |
| 视觉感知算法 | 指用于自动驾驶中的图像处理和分析算法，主要包括目标检测、物体跟踪等。 |
| 数据集格式统一 | 算法依据数据集的格式进行读取和处理，这意味着算法需要为不同格式的数据集设计不同的读取和处理方式。一旦数据集格式统一，算法就可以无缝适配不同的数据集，而无需进行修改。 |
| 模型训练 | 模型训练是通过对算法进行反复优化，使其能够根据输入数据进行预测或分类的过程。 |
| 训练配置 | 指用户在平台上设置和调整模型训练的参数，如设置学习率、迭代次数、批次大小、权重衰退等。这些配置直接影响模型训练的效果和效率。 |

3. **项目约束**

3.1 **时间约束**

**短期开发约束(课程项目时间内)：**  
项目需要在短期内完成“一键训练”功能的核心流程开发，包括以下内容：

1. **数据集与算法准备：** 初步完成部分数据集格式的统一和和算法的适配，使其能够支持当前功能的实现需求。
2. **前后端实现：** 构建基本的前后端交互界面，实现“一键训练”功能的端到端操作。
3. **训练服务器搭建：** 搭建并配置训练服务器，满足模型训练的硬件和环境要求。

**长期开发约束：**  
随着项目的不断迭代，需满足以下扩展性需求：

1. **数据集和算法扩展：** 支持不断增加的数据集和适配的算法，以满足新的应用需求和模型性能优化的目标。
2. **功能完善与扩展：** 在“一键训练”功能基础上，不断迭代和优化功能。例如，未来可能增加在线标注工具等，进一步提升系统的使用便捷性和功能完整度。

3.2 **资源约束**

项目开发受限于当前可用资源，包括：

1. **硬件资源：** 服务器性能（如CPU、GPU、内存等）必须满足模型训练和运行需求。
2. **人力资源：** 团队成员的数量及技术能力对开发效率和实现深度有直接影响，需合理分配任务。
3. **时间资源：** 开发时间有限，需要优先实现核心功能，确保按计划交付。

3.3 **质量约束**

项目需满足以下质量要求：

1. **功能完整性：** 确保用户从数据集获取到模型训练的“一键训练”功能流程的正确性和可靠性。
2. **性能要求：** 通过使用高性能的服务器使训练过程的耗时和系统响应速度需在合理范围内，满足实际使用需求。
3. **容错性：** 在训练过程中出现异常时，需具备相应的错误提示和处理机制，避免系统崩溃。
4. **可维护性：** 代码结构需清晰、模块化，为后续迭代和扩展提供便利。

3.4 **用户约束**

**用户类型：** 项目的目标用户包括技术开发者（如数据科学家、算法工程师）和普通业务用户（AI初学者）。

**使用门槛：** 系统需兼顾易用性和专业性，既要保证普通用户能通过简单操作完成训练流程，又要保证技术用户能灵活调整参数。

**交互需求：** 前端界面设计需直观友好，用户操作需有明确的信息反馈。

3.5 **解决方案约束**

**工具和技术：**

* **后端：** Spring Boot（版本：3.1.2），用于搭建后端服务框架，提供高性能和稳定性支持。
* **前端：** Vue 3（版本：3.3.4），用于构建用户界面，具备组件化和高可维护性。
* **数据库：** MySQL（版本：8.0.32），支持高效数据存储和查询操作。
* **深度学习框架：** PyTorch（版本：2.0），因其强大的生态系统和易用性，适用于模型训练和部署。

**技术选择原因：**

* **成熟度：** 所选技术均为成熟框架，社区活跃度高，便于获取支持和资源。
* **兼容性：** 选用技术彼此兼容，能有效提高开发效率和系统稳定性。
* **性能需求：** 满足项目对高效开发和运行性能的需求。

3.6 **外部系统约束**

项目需与以下外部系统集成：

1. **数据集和算法管理系统：** 本项目仅作为数据集和算法的使用方，数据集和算法管理系统负责存储、管理和提供各类数据集（如 KITTI、nuScenes）和算法（如 FAST R-CNN、SSD），为本项目实现训练任务提供基础输入。（**虽然数据集和算法管理属于外部系统，但是数据集格式的统一和算法的适配工作是小组独立完成的。**）
2. **高性能算力服务器：** 高性能算力服务器为项目提供模型训练所需的计算资源。项目会将数据和算法部署到该服务器，并通过任务请求的形式触发训练。服务器接收任务请求后执行训练操作，并返回训练结果（如模型文件等）。

3.7 **预期工作环境约束**

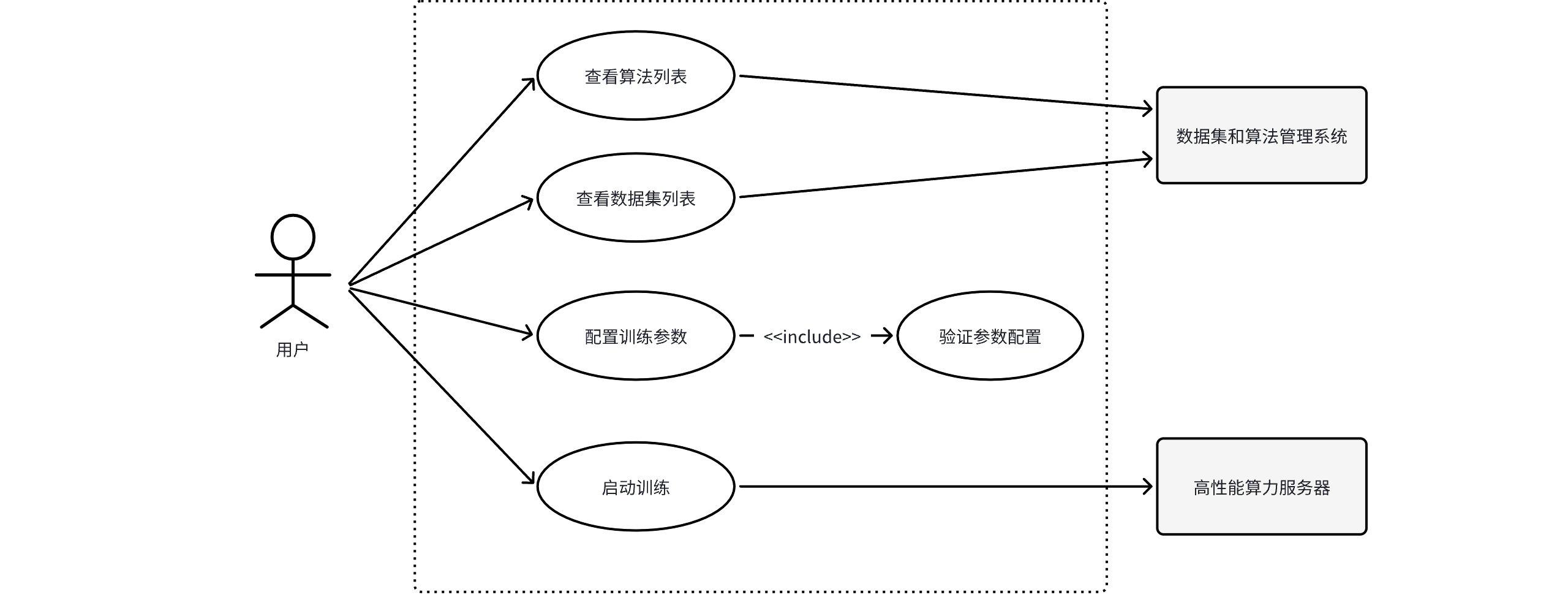
**开发环境：**

* 操作系统：Windows 10 / Ubuntu 20.04
* 开发工具：IntelliJ IDEA（后端开发）、VS Code
* 部署服务器：Linux系统，需支持Docker容器化部署。
* 网络环境：支持稳定的互联网连接，保证与外部系统的交互正常。

4. **功能性需求**

4.1 **训练配置模块**

4.1.1 **基于场景的建模**



**用例分析**

**用户**：负责与系统进行交互，执行数据和算法的查询、训练参数配置和训练任务启动。

**目标：**用户能够顺利查看算法与数据集列表、配置训练参数、启动训练，并与外部数据集和算法管理系统集成完成任务。

**前置条件**

1. 用户已登录系统。
2. 数据集和算法管理系统正常运行，可供查询和访问。
3. 高性能算力服务器处于可用状态。

**后置条件**

1. 用户完成算法与数据集的查询任务。
2. 用户成功配置训练参数并验证配置。
3. 启动训练后，系统将请求发送至算力服务器，任务开始执行。

**触发器**

用户登录系统，开始执行查询、参数配置或训练任务。

**基本流程**

1. **查看算法列表**

用户选择查看算法列表，系统通过外部“数据集和算法管理系统”返回算法列表。

1. **查看数据集列表**

用户选择查看数据集列表，系统通过外部“数据集和算法管理系统”返回数据集列表。

1. **配置训练参数**

用户选择配置训练参数。

提交后系统验证参数配置，确保用户输入的参数符合要求。

1. **启动训练**

用户选择启动训练任务。

系统根据配置的参数，将训练任务提交到高性能算力服务器，并等待返回结果。

**扩展流程**

1. **配置训练参数时验证失败**

系统提示用户参数输入不合法或缺失。

用户重新修改参数配置。

1. **外部系统不可用**

当“数据集和算法管理系统”或高性能算力服务器不可用时，系统提示错误信息。

**异常**

1. **数据集与算法管理系统异常**

系统无法返回算法或数据集列表，提示用户稍后重试或联系管理员。

1. **算力服务器资源不足**

训练任务提交失败，系统返回错误信息，并建议用户稍后重试。

1. **用户配置参数错误**

系统在验证参数时发现错误，阻止训练任务启动。

**优先级：**高

**何时可用：**第一次增量

**使用频率：**中等到高频。

**使用方式：**

用户通过界面执行各个功能，包括：

查询数据集和算法列表

配置训练参数

启动训练任务

**次要参与者：**

数据集和算法管理系统

高性能算力服务器

**次要参与者使用方式**

1. **数据集和算法管理系统**：提供数据集和算法信息供用户查询。
2. **高性能算力服务器**：执行用户提交的训练任务，并返回训练结果。

4.1.2 **基于类的建模**

图示

描述已自动生成

User类通过其UserID属性与StartTraining类建立关联关系，表示每个用户可以启动训练任务。用户通过调用 StartTraining类的StartTraining(userID,ConfigurationID)方法，将任务与自身账户绑定，从而实现任务的身份关联和管理。

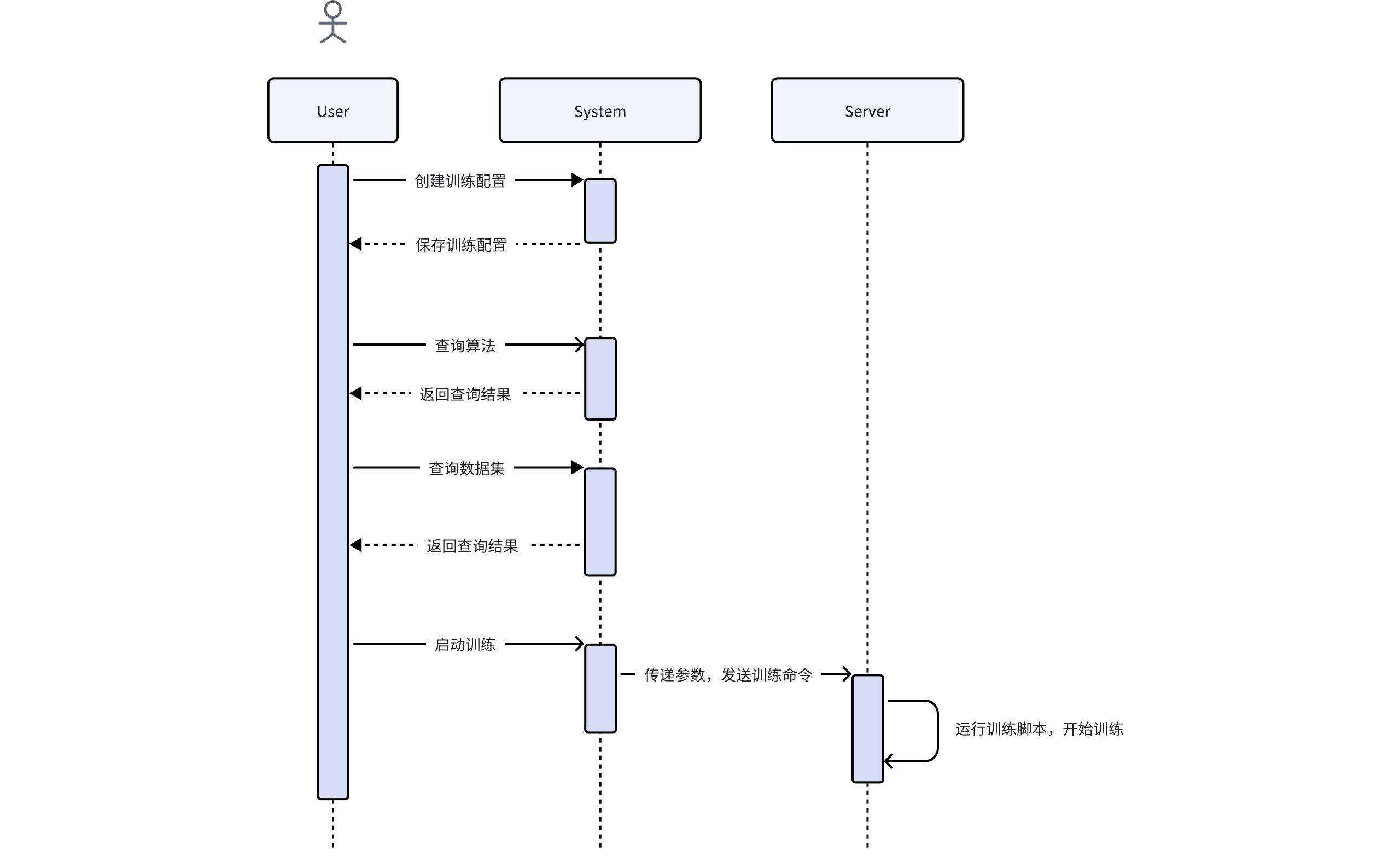
AlgorithmDatasetSystem 类与 CreateConfiguration 类之间存在依赖关系。AlgorithmDatasetSystem 通过其 getAlgorithmList() 和 getDatasetList() 方法提供算法列表和数据集列表，用于支持 CreateConfiguration 创建具体的训练配置。这表明训练配置的创建依赖于 AlgorithmDatasetSystem 提供的算法和数据集信息，是其实现的基础。

CreateConfiguration 类与 TrainingParams 类通过依赖关系紧密关联。TrainingParams 提供了具体的训练参数，包括学习率、训练轮数、批量大小等细节，作为训练配置的重要组成部分。而 CreateConfiguration 使用这些参数生成完整的训练配置，为后续任务执行提供明确的指导。

在训练任务启动方面，StartTraining 类依赖于 CreateConfiguration 类的配置，通过传入 ConfigurationID 来指定所需的训练配置。在启动训练任务时，必须基于已有的训练配置执行任务调度与处理。

此外，CreateConfiguration 类与 HighPerformanceServer 类之间存在关联关系。在创建训练配置时，系统需要调用 HighPerformanceServer 的 getHighPerformanceServer() 方法来获取高性能服务器资源的标识。这些服务器资源为训练任务的执行提供了必要的计算能力，确保系统的高效运行。

4.1.3 **基于行为的建模**



这张时序图展示了用户通过系统完成训练任务的完整流程，涉及用户、算法与数据集管理系统（AlgorithmDatasetSystem）以及高性能服务器（HighPerformanceServer）之间的交互。整个流程可以分为四个主要阶段。

首先，用户通过界面创建训练配置。在这一阶段，用户输入训练所需的参数，例如学习率、训练轮数、批量大小等。系统接收用户的输入后，将这些参数保存为训练配置，确保后续任务能够基于该配置顺利执行。这一步是整个流程的起点，用户的输入决定了后续训练任务的具体内容。

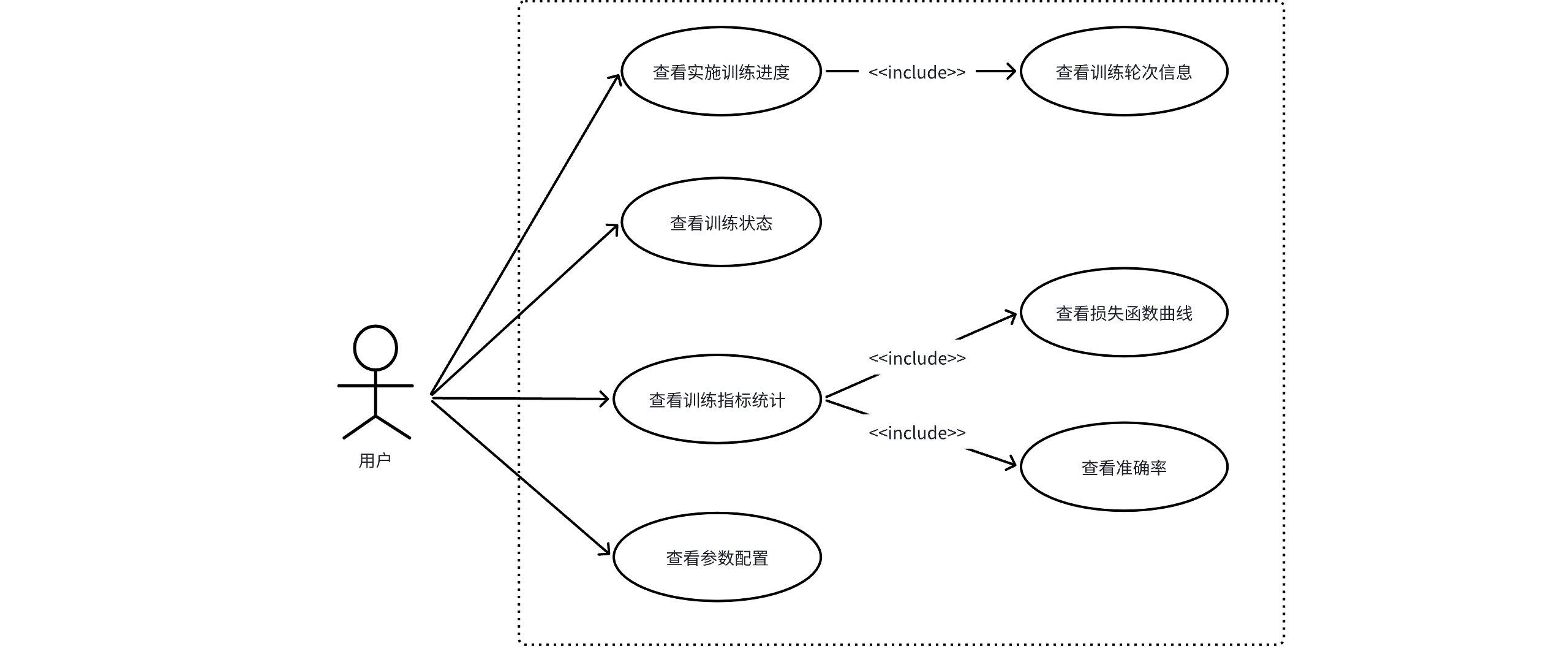
接下来，用户需要查询算法和数据集列表。用户选择查询算法时，系统会向外部的“算法与数据集管理系统”发送请求，获取可用的算法列表，并将结果返回给用户。同样，当用户选择查询数据集时，系统会再次向“算法与数据集管理系统”发送请求，获取可用的数据集列表，并将结果展示给用户。这两个查询操作为用户提供了选择的依据，确保用户能够根据实际需求选择合适的算法和数据集。

在完成训练配置和查询操作后，用户可以启动训练任务。用户选择启动训练时，系统会将训练配置中的参数（包括算法、数据集和训练参数）传递给高性能服务器，并发送训练命令。高性能服务器接收到这些信息后，会运行相应的训练脚本，正式开始训练任务的执行。这一阶段是整个流程的核心，系统通过与高性能服务器的交互，完成了从配置到实际训练的过渡。

最后，高性能服务器开始执行训练任务，用户等待训练结果的返回。整个流程通过用户、系统和外部组件的协作，完成了从训练配置到任务执行的完整闭环。

4.2 **训练监控模块**

4.2.1 **基于场景的建模**



**用例分析**

**用户**：负责执行各项查询操作，了解训练过程、状态、参数和结果。

**目标：**用户可以查询训练的进度、状态、统计指标、参数配置及日志，全面监控训练过程。

**前置条件**

1. 用户已登录系统。
2. 训练任务已启动，或相关数据（状态、日志、指标）可供查询。

**后置条件**

1. 用户完成对训练信息的查询任务。
2. 查询结果正确展示给用户。

**触发器**

用户选择需要查询的功能，如“查看训练状态”“查看训练指标统计”等。

**基本功能**

1. **查看实施训练进度**

用户查看当前训练任务的整体进度信息。

1. **查看训练状态**

用户获取当前训练任务的执行状态（如运行中、已完成、失败等）。

1. **查看训练指标统计**

用户查询训练过程中生成的统计指标，包括损失函数曲线与准确率。

1. **查看参数配置**

用户查询训练任务的参数配置，如学习率、批次大小、训练轮次等关键信息。

1. **查看训练日志**

用户查询训练任务生成的执行日志，以排查问题或了解详细过程。

**异常**

1. **系统超时或无响应**

系统给出相应提示，用户稍后重试。

**优先级：**高。

**何时可用：**系统部署完成。

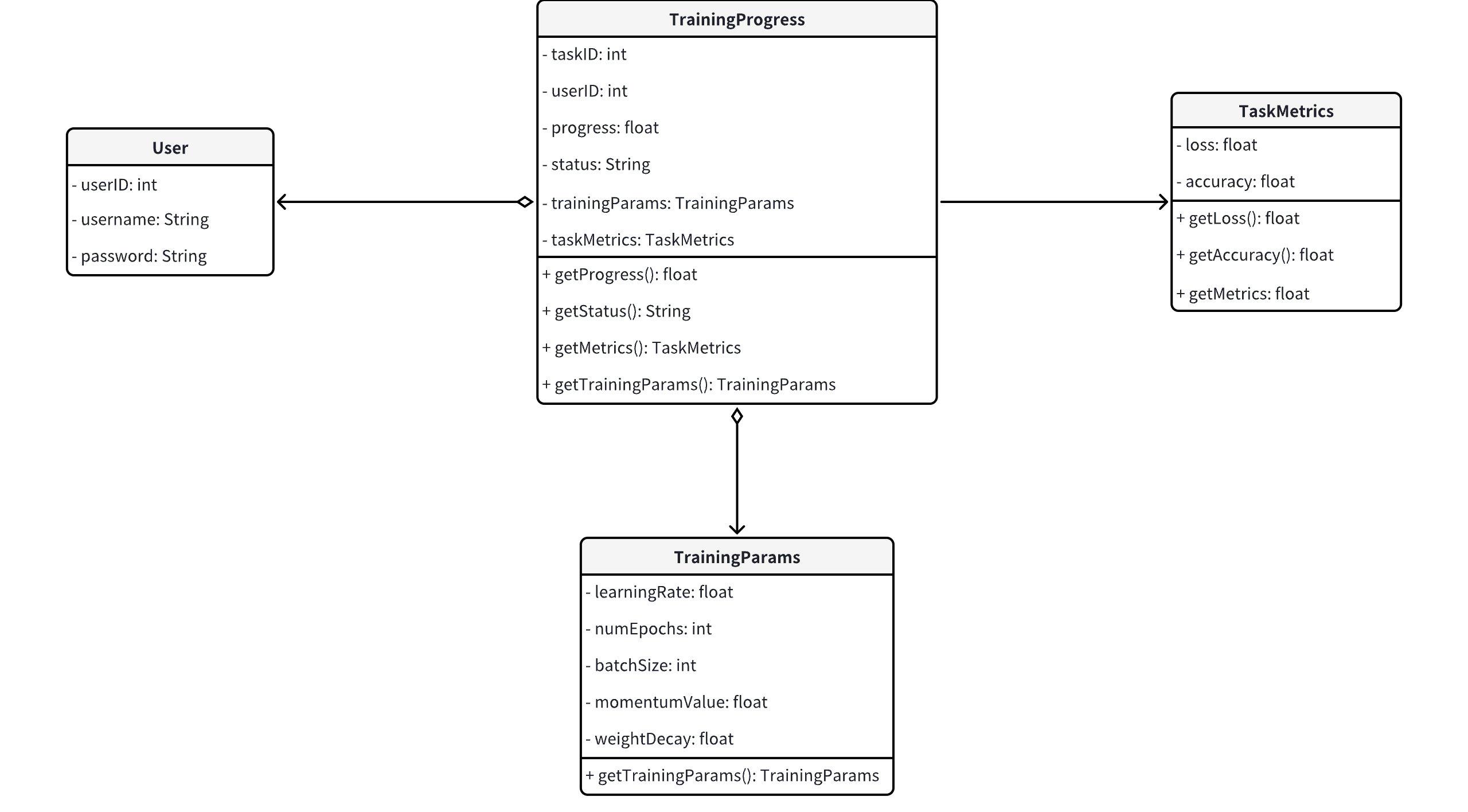
**使用频率：**高。用户监控训练过程时频繁使用。

**使用方式：**

用户通过系统界面选择相应的查询功能（如：查看状态、指标等）。

系统返回查询结果，并以图表或文本形式展示。

4.2.2 **基于类的建模**

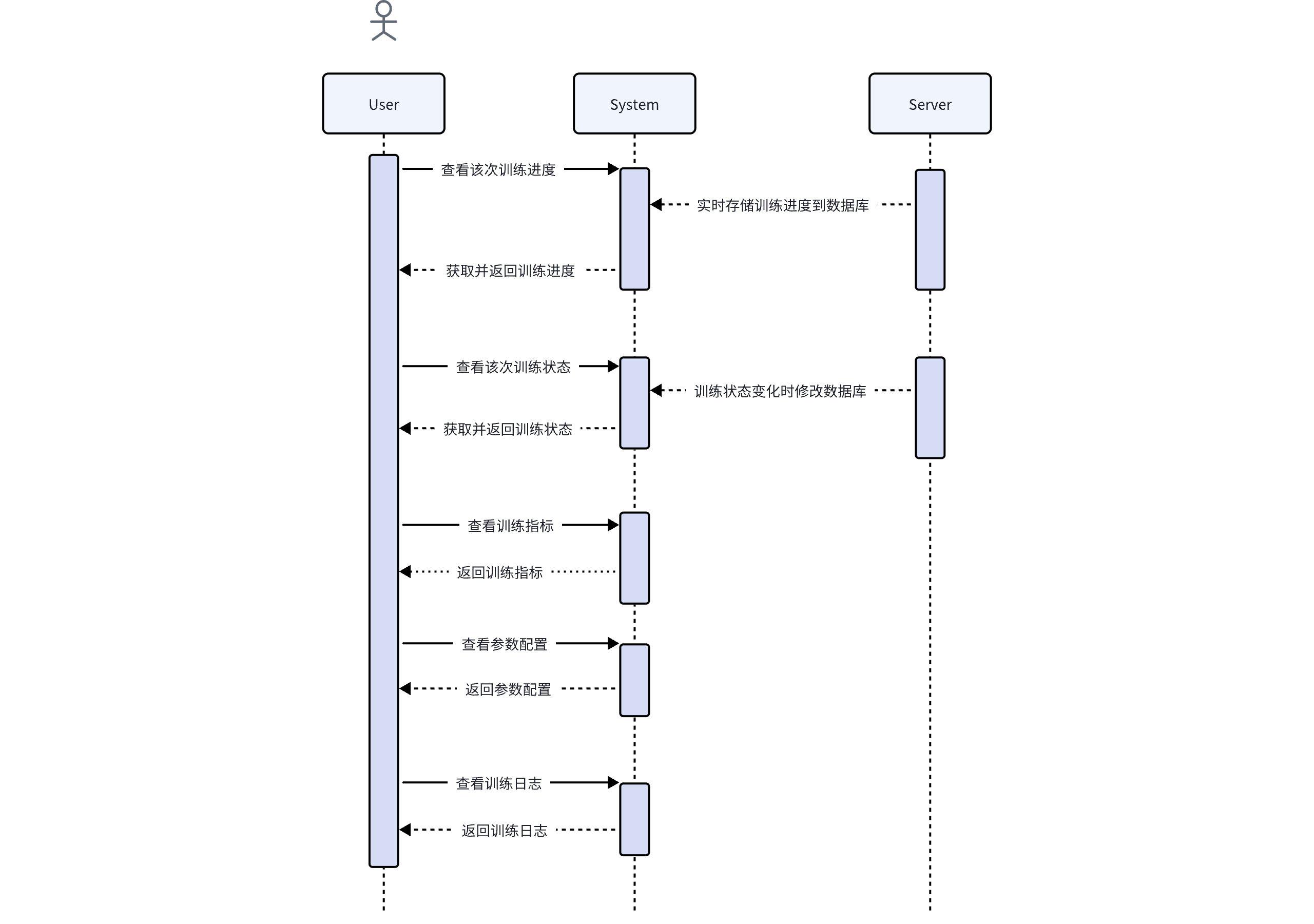


User 类和 TrainingProgress 类之间存在组合关系。具体而言，通过 userID 建立连接，一个用户可以拥有多个训练任务的进度信息。这种关系表示一个用户可以管理多个训练任务的进展。

TrainingProgress 类与 TrainingParams 类之间存在组合关系。TrainingProgress 包含了一个 TrainingParams 对象，表示每个训练任务进度都必须包含其对应的训练参数配置。这种组合关系强调了训练参数是训练任务不可分割的一部分，二者生命周期紧密绑定。

TrainingProgress 类与 TaskMetrics 类之间是关联关系。TrainingProgress 类通过调用 getMetrics() 方法依赖 TaskMetrics 类提供的性能指标（如损失值和准确率）。这种关系表明，训练任务的进度需要借助 TaskMetrics 来获取相关的训练性能数据，但两者的生命周期是独立的。

4.2.3 **基于行为的建模**



这张时序图描述了用户通过系统监控训练任务的各类查询操作，涉及用户、系统以及服务器之间的交互。以下是对时序图中各部分流程的详细描述：

用户首先选择查看训练任务的进度。系统接收到用户请求后，会从实时存储的训练进度数据中获取相关信息。这些数据通常由服务器在任务执行过程中不断更新并存储到数据库中。系统处理完查询后，将训练进度信息返回给用户，用户可以在界面上查看任务的当前进度。

用户选择查看训练任务的状态，例如“运行中”、“已完成”或“失败”等。系统接收到请求后，会从数据库中检索训练状态信息。训练状态的更新通常由服务器在任务状态发生变化时写入数据库。系统将查询结果返回给用户，用户可以在界面上直观了解训练任务的当前状态。

用户选择查询训练过程中生成的统计指标，例如损失函数曲线或准确率等。系统接收到请求后，会从数据库中获取相关的训练指标数据。这些指标通常由服务器在训练过程中实时计算并存储。系统将训练指标以图表或文本形式返回给用户，帮助用户评估训练的效果。

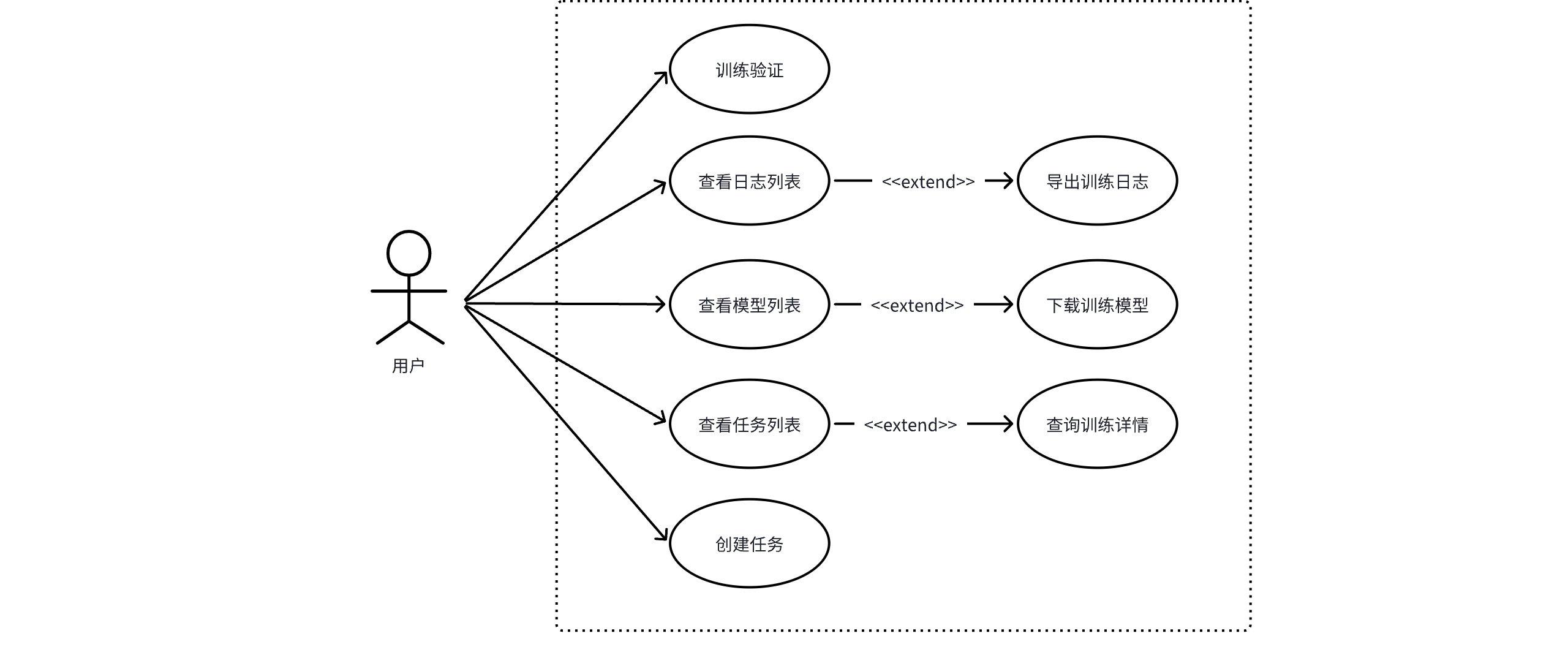
用户选择查看训练任务的参数配置，例如学习率、批量大小或训练轮次等。系统接收到请求后，从数据库中检索对应的参数配置。这些参数通常在训练任务启动时由用户定义并存储。系统将参数配置返回给用户，用户可以核对训练任务的具体设定。

用户选择查询训练任务的日志信息，用于排查问题或了解训练的详细执行过程。系统接收到请求后，从数据库中获取日志数据。这些日志通常由服务器在训练过程中生成并存储。系统将日志内容返回给用户，用户可以在界面上查看详细的训练过程记录。

总体而言，用户通过界面发起查询请求，系统从数据库中检索相关信息，这些数据由服务器在任务执行过程中实时更新并存储。系统将查询结果返回给用户，帮助用户全面了解训练任务的进展、状态、性能指标、参数配置及日志信息。

4.3 **任务管理模块**

4.3.1 **基于场景的建模**



**用例分析**

**用户**：查询训练任务的日志、模型与任务列表，进一步导出、下载或查询详细记录。

**目标：**用户高效管理训练任务，查询训练日志、模型、任务列表，并执行进一步操作（如导出日志、下载模型）。

**前置条件**

1. 用户已登录系统。
2. 系统中存在训练日志、模型与任务列表数据。

**后置条件**

1. 用户成功执行查询操作，系统返回相应数据或文件。

**触发器**

用户选择“查看日志列表”“查看模型列表”或“查看任务列表”功能。

**基本功能**

1. **查看日志列表**

用户选择“查看日志列表”，系统展示训练日志的简要信息。

1. **查看模型列表**

用户选择“查看模型列表”，系统展示训练生成的模型信息。

1. **查看任务列表**

用户选择“查看任务列表”，系统展示当前任务的基本信息。

1. **创建任务**

用户选择“创建任务”，系统将跳转训练配置模块进行任务配置。

1. **训练验证**

用户选择“训练验证”，系统将对训练结束的模型进行验证。

**异常**

1. **导出训练日志时文件生成失败**

系统给出相应提示，用户稍后重试。

**优先级：**高

**何时可用：**系统部署完成。

**使用频率：**

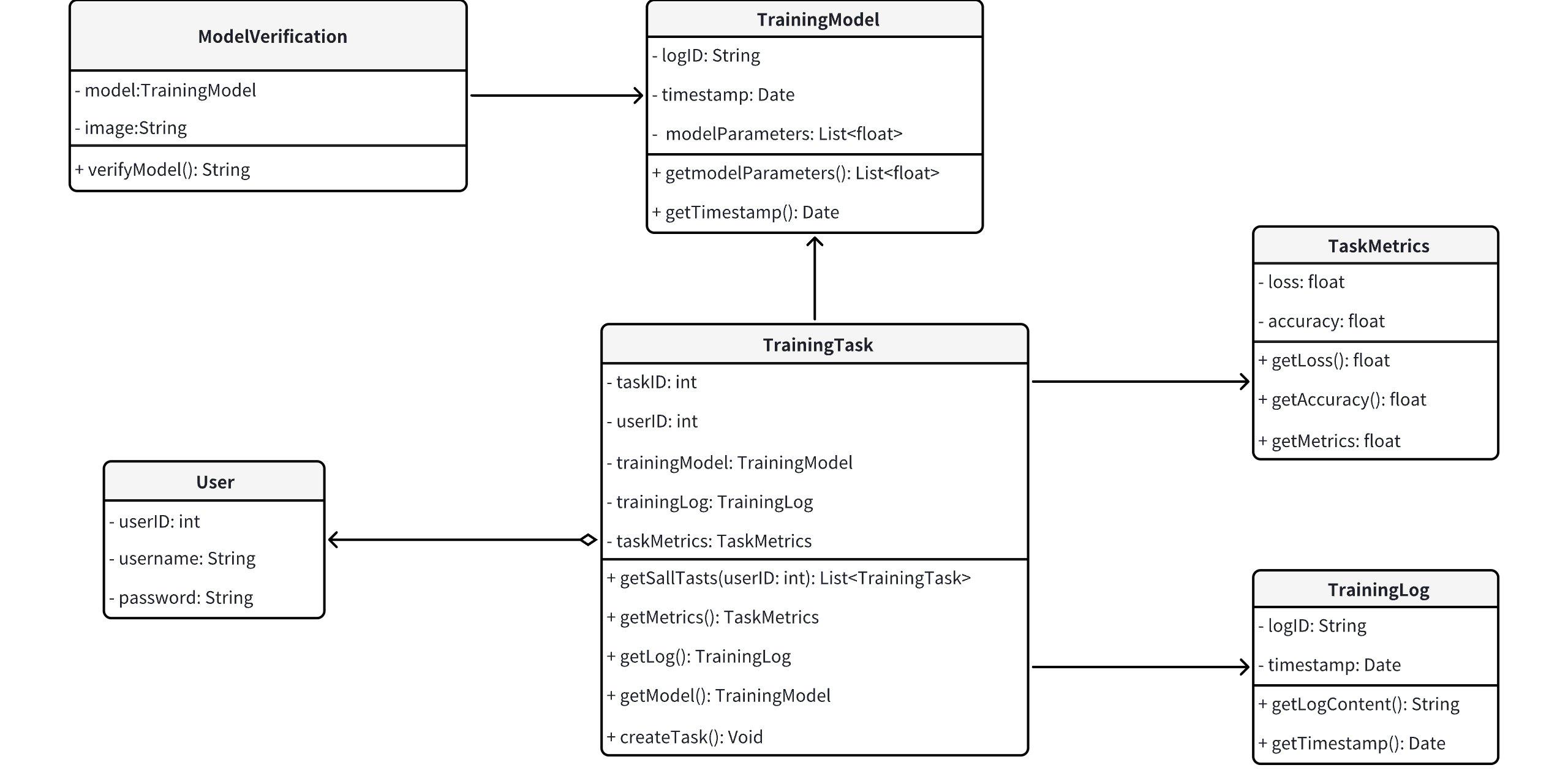
高频：用户实时查看日志与任务列表，掌握训练任务进度。

中频：用户在任务结束后下载模型与导出日志。

**使用方式：**

1. 用户通过系统界面选择查询功能（如“查看日志列表”）。
2. 系统展示对应信息，并根据用户操作提供导出下载功能。

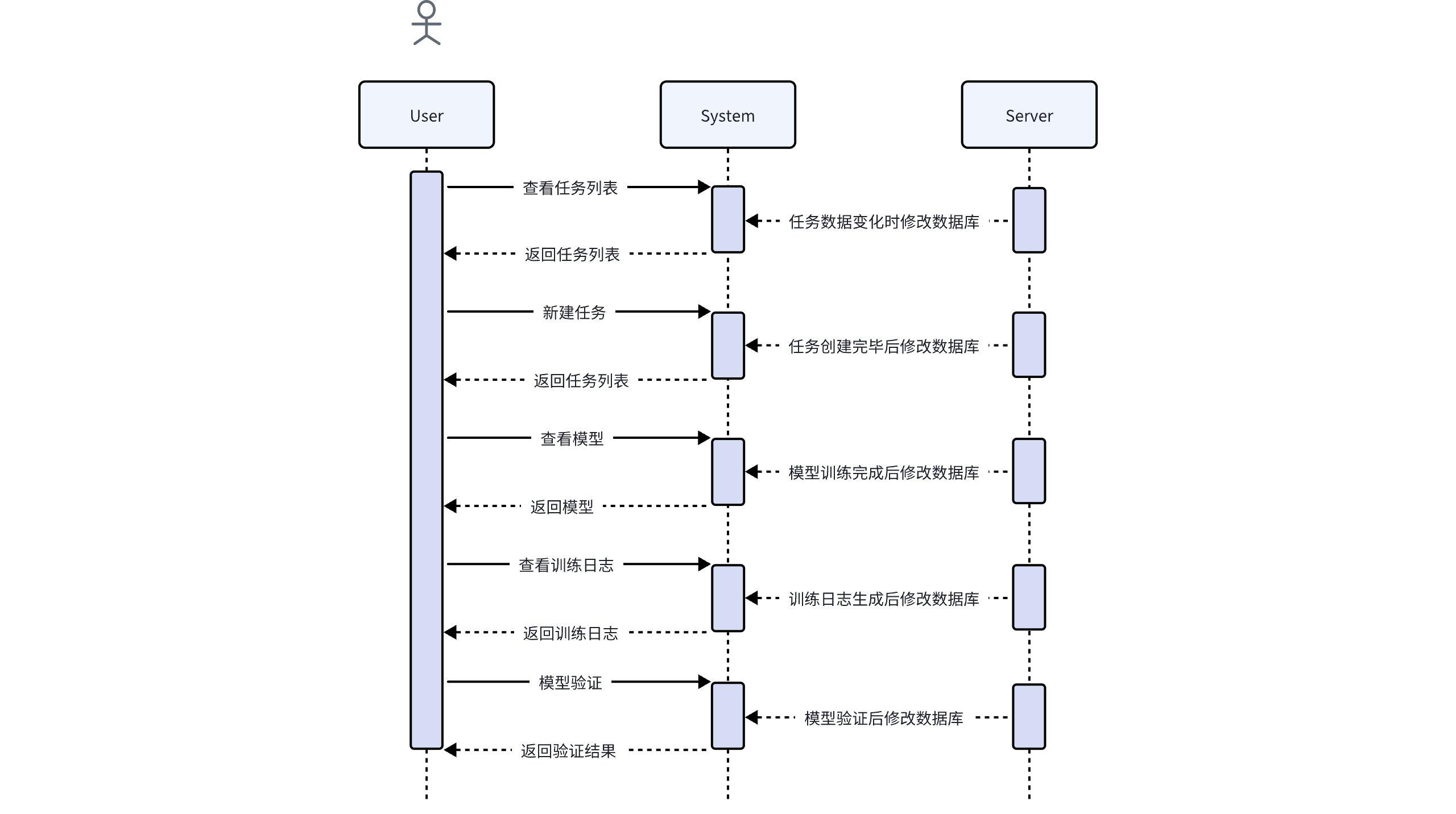
4.3.2 **基于类的建模**



User 类与 TrainingTask 类通过 userID 进行关联。每个用户都可以查询训练任务，训练任务通过记录 userID 来标识属于特定用户的任务。这种关系体现了用户对训练任务的管理功能。  
 TrainingTask 类与 TrainingLog 类通过关联关系相连。每个训练任务会生成对应的训练日志，记录任务的详细执行过程，包括时间戳和日志内容。TrainingTask 通过 getLog 方法与 TrainingLog 建立联系，为任务的可追溯性和调试提供支持。  
 TrainingTask 类与 TrainingModel 类通过关联关系相连。每个训练任务完成后都会生成一个具体的训练模型，包括模型参数和时间戳等信息。TrainingTask 通过 getModel 方法获取 TrainingModel，为后续模型使用和评估提供支持。

ModelVerification类于TrainingModel类通过关联关系相连。每个训练任务完成后ModelVerification类都会通过verifyModel方法对训练好的模型进行验证并生成验证图片。

4.3.3 **基于行为的建模**



这张时序图描述了用户在系统中管理训练任务的各类操作过程，包括查询任务日志、模型、任务列表，以及进一步的导出、下载和训练验证等操作。以下是对时序图中各部分流程的详细描述：

1. **查看任务列表**  
   用户选择“查看任务列表”功能，系统接收到请求后，会从数据库中检索所有与该用户相关的训练任务的基本信息，包括任务名称、状态和创建时间等。检索完成后，系统将任务列表返回给用户，用户可以在界面上查看并筛选感兴趣的任务。
2. **查看日志列表**  
   用户选择“查看日志列表”功能，系统从数据库中提取与训练任务相关的日志信息，例如任务的执行时间、错误提示及过程记录等。日志数据经过处理后展示给用户，用户可以选择进一步导出日志文件。如果导出失败，系统会提示用户稍后重试。
3. **查看模型列表**  
   用户选择“查看模型列表”功能，系统从数据库中加载所有已完成训练任务生成的模型信息，包括模型名称、参数规模及创建时间等。检索完成后，系统将模型列表返回给用户，并提供模型下载功能。用户可选择感兴趣的模型进行下载。
4. **创建任务**  
   用户选择“创建任务”功能，系统将跳转到训练配置模块，用户在该模块中设置训练参数（如算法、数据集及资源配置）。完成配置后，系统会生成新的任务，并将其加入任务列表。
5. **训练验证**  
   用户选择“训练验证”功能，系统将对指定任务生成的模型进行验证。验证过程由 ModelVerification 类触发，其内部调用验证方法 verifyModel，并生成一组验证结果（如性能指标或验证图片）。验证完成后，系统将结果返回给用户，供其评估模型的质量。
6. **导出日志或下载模型**  
   用户选择导出日志或下载模型，系统会根据用户选择的具体项从数据库中生成文件。若文件生成成功，系统将文件提供给用户下载；若生成失败，系统会提示用户稍后重试。

5. **非功能性需求**

5.1 **可用性需求**

系统操作需便捷，用户能够快速理解功能，无需复杂学习即可上手使用。为此，系统需提供清晰直观的导航菜单和操作指引，设计简洁的输入界面，减少不必要的操作步骤。界面友好性也至关重要，系统界面应简洁美观，并通过响应式设计适配不同设备与分辨率，确保用户获得良好的使用体验。

5.2 **兼容性需求**

系统需兼容主流操作系统，包括 Windows和macOS ，前端开发基于跨平台框架Vue.js，后端服务部署在兼容性强的环境（如 Docker 容器）。此外，系统需兼容主流浏览器（如 Chrome、Edge等），采用标准的 HTML5 和 CSS3 技术，避免过时特性，并通过浏览器兼容性测试工具进行全面验证。

5.3 **可维护性需求**

系统需具备良好的代码可读性，代码需清晰易懂，遵循统一的代码风格指南，合理添加注释与文档，确保易于维护。系统设计应具备较高的模块化程度，采用分层架构（微服务架构），将功能划分为独立模块以降低耦合度。此外，系统需进行版本控制，使用 Git 进行代码管理，以实现快速且可控的版本迭代。

5.4 **可扩展性需求**

系统应具备良好的功能扩展性，以支持后续功能的添加。系统需采用微服务架构，便于新增功能模块；在数据库设计中预留扩展字段，以支持动态扩展数据结构；通过前后端解耦，支持不同功能模块的独立开发与部署。系统在功能扩展时需确保平滑升级，不影响现有功能的正常运行。

6. **问题探讨**

6.1 **为什么数据和算法管理部分的数据集转化和算法适配是小组完成，但要作为外部系统？**

这是因为将数据集转换为统一格式的过程无法完全通过自动化脚本实现。一个脚本通常只能处理特定类型的数据集，同样，算法也需要根据统一的数据集进行针对性的调整。因此，当系统中需要添加新的数据集或算法时，系统维护人员需要在规定时间内进行实时处理，确保系统的正常运行和维护。然而，由于当前人力资源有限，我们仅实现了两个数据集格式的转换和两个算法的适配，无法长期维护用户上传的所有算法和数据集。未来，当人力和时间条件允许时，我们可以进一步将数据集和算法管理功能整合到项目系统中，而并不是作为外部系统，实现更高程度的自动化和维护效率。

6.2 **为什么可以保证算法读取原始数据集和统一数据集的数据集是相同的？**

**首先**，在数据集转换过程中，我们严格遵循数据格式转换的规范和规则，确保数据的完整性和准确性不受影响。例如，对于一个包含图像和标签的原始数据集（如 COCO 格式），我们将其转换为 Pascal VOC 格式时，仅调整了标签的存储格式（如由 JSON 转为 XML），但不会修改图像内容或标签值，数据的核心信息保持不变。

(下面代码仅简单说明统一数据集不会导致数据的丢失，和相同算法使用原始数据集获取的信息一致)

|  |
| --- |
| '''原始数据集格式''' {"images": [{"id": 1,"file\_name": "image1.jpg","width": 640,"height": 480}], "annotations": [{"image\_id": 1,"category\_id": 3,"bbox": [100, 150, 200, 200]}], "categories": [{"id": 3,"name": "dog"}]}  '''统一数据集格式''' <annotation>  <filename>image1.jpg</filename>  <size>  <width>640</width>  <height>480</height>  </size>  <object>  <name>dog</name>  <bndbox>  <xmin>100</xmin>  <ymin>150</ymin>  <xmax>300</xmax>  <ymax>350</ymax>  </bndbox>  </object> </annotation>   '''原始算法使用该部分数据''' # 加载 COCO 数据集 from pycocotools.coco import COCO coco = COCO('annotations/instances\_train2017.json') # 获取图像信息 image\_info = coco.loadImgs(1)[0] # 获取标签信息 annotations = coco.loadAnns(coco.getAnnIds(imgIds=image\_info['id'])) # 输出标签信息 print(annotations)   '''调整后算法使用这部分数据''' # 解析 Pascal VOC 格式的 XML 文件  import xml.etree.ElementTree as ET  # 获取图像和标签信息  tree = ET.parse('annotations/VOC2007/Annotations/image1.xml')  root = tree.getroot()  filename = root.find('filename').text  object\_name = root.find('.//object/name').text  bbox = root.find('.//bndbox') xmin = int(bbox.find('xmin').text) ymin = int(bbox.find('ymin').text)  xmax = int(bbox.find('xmax').text)  ymax = int(bbox.find('ymax').text)  print(f"Filename: {filename}, Object: {object\_name}, BBox: {xmin}, {ymin}, {xmax}, {ymax}") |

**其次**，数据的表示方式可能会发生变化，但是其核心数据信息不会发生变化。例如，旋转信息可以从四元数表示转换为轴角表示，这种转换仅涉及存储格式的调整，核心数据本身（如旋转信息）保持不变，确保转换后的数据与原始数据具有相同的准确性和完整性。

(下面代码仅简单说明统一数据集不会导致数据的丢失，和相同算法使用原始数据集获取的信息一致)

|  |
| --- |
| '''原始数据集格式''' {  "rotation": {  "w": 0.7071,  "x": 0.7071,  "y": 0,  "z": 0  } } '''统一数据集格式''' {  "rotation\_axis": {  "x": 1,  "y": 0,  "z": 0  },  "rotation\_angle": 1.5708 } '''现在要实现向量的旋转''' '''原始算法使用该部分数据''' import numpy as np from scipy.spatial.transform import Rotation as R # 使用四元数表示旋转 q = [0.7071, 0.7071, 0, 0]  r = R.from\_quat(q) # 通过四元数旋转对象 vector = np.array([1, 0, 0])  rotated\_vector = r.apply(vector) # 输出旋转后的向量 print(rotated\_vector)  # 输出：[1. 0. 0.]  '''调整后算法使用这部分数据''' import numpy as np from scipy.spatial.transform import Rotation as R # 使用轴角表示旋转 axis = [1, 0, 0] # 旋转轴 (x, y, z) angle = 1.5708 # 旋转角度 90 度（以弧度表示） # 通过旋转轴和角度旋转对象 r = R.from\_rotvec(axis \* angle)  # 旋转向量 vector = np.array([1, 0, 0])  rotated\_vector = r.apply(vector)  print(rotated\_vector)  # 输出：[1. 0. 0.] |

7. **参考文献**

[1] Robertson, S., & Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*

(3rd ed.). Addison-Wesley.