AASD - Zajęcia Zintegrowane 2

Temat: System zarządzania schroniskiem dla zwierząt

Zespół MAJ:

- Michał Laskowski 310181
- Adam Dąbkowski 310035
- Jarosław (Yaroslav) Harbar 317044

GitHub

Link do publicznego repozytorium: https://github.com/kenjakendi/AASD-MAJ

Wymagania funkcjonalne:

- Rejestr zwierząt - system zapewnia prowadzenie cyfrowej ewidencji zwierząt z możliwością dodawania, edycji oraz śledzenia stanów zdrowia, statusów (np. "nakarmiony", "na spacerze") i historii zdarzeń.
- Rejestr pracowników i ról - system zapewnia możliwość zarządzania kontami i rolami pracowników (opiekun, sprzątacz, weterynarz) wraz z przypisaniem kompetencji, limitów i preferencji.
- Dostępność personelu - system umożliwia zgłaszanie i aktualizowanie slotów dostępności przez pracowników.
- Generowanie zadań opieki - system automatycznie generuje zadania takie jak karmienie, spacery, szczepienie czy sprzątanie (cyklicznie lub na żądanie).
- Automatyczny przydział zadań - system zapewnia automatyczne dopasowanie zadań do dostępnych pracowników na podstawie ich kompetencji, preferencji i aktualnego obciążenia.
- Monitorowanie cyklu życia zadania - system umożliwia śledzenie pełnego cyklu życia zadania oraz potwierdzanie wykonania przez pracowników.
- Kalendarz szczepień - system automatycznie planuje i monitoruje okna szczepień oraz generuje odpowiednie zadania dla personelu.
- Sprzątanie pomieszczeń - system automatycznie generuje zadania sprzątania dziennego i na żądanie na podstawie stanu pomieszczeń.
- Przyjęcia zwierząt - system zapewnia proces rejestracji nowych zwierząt, przeprowadzenie badań wstępnych oraz włączenie ich do harmonogramu opieki.
- Priorytetyzacja zadań - system automatycznie rozróżnia zadania krytyczne od rutynowych i dynamicznie aktualizuje priorytety w zależności od sytuacji.
- Sprawiedliwa rotacja opieki - system zapewnia mechanizm równomiernego rozdziału zadań między zwierzęta i opiekunów, dbając o sprawiedliwą rotację.
- Obsługa nagłych zdarzeń - system zapewnia natychmiastową rekonfigurację planu w przypadku nagłych zdarzeń, takich jak przyjęcie nowego zwierzęcia lub nieobecność wolontariusza.
- Wsparcie procesu adopcyjnego - system prowadzi proces adopcji, od złożenia wniosku po finalizację adopcji.

Model ról:

Role Schema: Koordynator (Coordinator)
Description: odpowiada za centralne planowanie, przydzielanie i monitorowanie realizacji podstawowych zadań oraz szczepień. Reorganizuje harmonogramy w przypadku zdarzeń nagłych, zmian dostępności personelu lub priorytetów.
Protocols and Activities: CleanRequest, AssignCleanTask, ConfirmCleanTask, FeedRequest, AssignFeedTask, ConfirmFeedTask, WalkRequest, AssignWalkTask, ConfirmWalkTask, AssignVaccinationTask, ConfirmVaccinationTask, AdoptionApplicationRequest, AdoptionApplicationTask, ConfirmAdoptionApplicationTask, HealthRequest, HealthRequestTask, ConfirmHealthRequestTask, RegisterNewAnimal, AssignInitialCheckupTask, ConfirmInitialCheckupTask, UpdateWorkerAvailabilityTask, ConfirmUpdateWorkerAvailabilityTask,
Permissions: generates taskId generates taskStatus reads newCleanRequest reads workerId reads workerAvailability reads roomState reads taskStatus reads newFeedRequest reads animalFeedState reads newWalkRequest reads animalWalkState reads newWorkerAvailability reads newAnimalRegistration reads animalHealthState reads newHealthRequest reads newAdoptionApplication reads animalAdoptionState
Responsibilities: Liveness: COORDINATOR = ((CleanRequest . AssignCleanTask . ConfirmCleanTask) | (FeedRequest . AssignFeedTask . ConfirmFeedTask) | (WalkRequest . AssignWalkTask . ConfirmWalkTask) | (AssignVaccinationTask . ConfirmVaccinationTask) | (AdoptionApplicationRequest . AdoptionApplicationTask . ConfirmAdoptionApplicationTask) | (HealthRequest . HealthRequestTask . ConfirmHealthRequestTask) | (RegisterNewAnimal . AssignInitialCheckupTask . ConfirmInitialCheckupTask) | (UpdateWorkerAvailabilityTask . ConfirmUpdateWorkerAvailabilityTask))*

Safety:
true

Role Schema: Opiekun (Caretaker)
Description: reprezentuje pracownika odpowiedzialnego za codzienną opiekę nad zwierzętami. Zgłasza swoją dostępność, realizuje przydzielone zadania takie jak karmienie czy spacery, a także potwierdza wykonanie powierzonych obowiązków.
Protocols and Activities: AssignFeedTask, ConfirmFeedTask, AssignWalkTask, ConfirmWalkTask, UpdateWorkerAvailabilityTask, ConfirmUpdateWorkerAvailabilityTask,
Permissions: generates newWorkerAvailability generates animalFeedState generates animalWalkState generates taskStatus reads animalId reads taskId reads taskStatus
Responsibilities: Liveness: CARETAKER = ((AssignFeedTask . ConfirmFeedTask) | (AssignWalkTask . ConfirmWalkTask) | (UpdateWorkerAvailabilityTask . ConfirmUpdateWorkerAvailabilityTask))* Safety: ∀ animalId, taskId: ConfirmFeedTask(taskId,animalId) ⇒ set(animalFeedState(animalId),"fed") ∧ set(taskStatus(taskId),"done"). ∀ animalId, taskId: ConfirmWalkTask(taskId,animalId) ⇒ set(animalWalkState(animalId),"walked") ∧ set(taskStatus(taskId),"done")

Role Schema: Sprzątacz (Cleaner)

Description: wykonuje zadania sprzątania dziennego oraz reaguje na zlecenia generowane na żądanie.

Protocols and Activities:
AssignCleanTask, ConfirmCleanTask,
UpdateWorkerAvailabilityTask, ConfirmUpdateWorkerAvailabilityTask,

Permissions:
generates newWorkerAvailability
generates roomState
generates taskStatus
reads roomId
reads taskId
reads taskStatus

Responsibilities:

Liveness:
CLEANER = (
  (AssignCleanTask . ConfirmCleanTask) |
  (UpdateWorkerAvailabilityTask . ConfirmUpdateWorkerAvailabilityTask)
)*

Safety:
∀ roomId,taskId: ConfirmCleanTask(taskId,roomId) ⇒ set(roomState(roomId),"clean") ∧ set(taskStatus(taskId),"done")

Role Schema: Weterynarz (Veterinarian)

Description: realizuje zadania związane z opieką medyczną (w tym szczepień).

Protocols and Activities:
AssignVaccinationTask, ConfirmVaccinationTask
HealthRequestTask, ConfirmHealthRequestTask
AssignInitialCheckupTask, ConfirmInitialCheckupTask
UpdateWorkerAvailabilityTask, ConfirmUpdateWorkerAvailabilityTask

Permissions:
generates newWorkerAvailability
generates taskStatus
generates animalHealthState
reads animalId
reads taskId
reads taskStatus

Responsibilities:

Liveness:

VETERINARIAN = (
  (AssignVaccinationTask . ConfirmVaccinationTask) |
  (AssignInitialCheckupTask . ConfirmInitialCheckupTask) |
  (HealthRequestTask . ConfirmHealthRequestTask) |
  (UpdateWorkerAvailabilityTask . ConfirmUpdateWorkerAvailabilityTask)
)*

Safety:

ConfirmVaccinationTask(taskId) ⇒ set(taskStatus(taskId),"done")

ConfirmHealthRequestTask(taskId,animalId) ⇒ set(animalHealthState(animalId), "healthy") ∧ set(taskStatus(taskId),"done")

ConfirmInitialCheckupTask(taskId,animalId) ⇒ set(animalHealthState(animalId),"healthy") ∧ set(taskStatus(taskId),"done")

Role Schema: Asystent Zwierzęcia (Animal Assistant)

Description: reprezentuje indywidualnego podopiecznego schroniska. Monitoruje stan zwierzęcia (np. głód, potrzeba ruchu) i emituje żądania opieki.

Protocols and Activities:
FeedRequest, ConfirmFeedTask
WalkRequest, ConfirmWalkTask
HealthRequest, ConfirmHealthRequestTask
ConfirmAdoptionApplicationTask
ConfirmInitialCheckupTask

Permissions:
generates newFeedRequest
generates newWalkRequest
generates newHealthRequest
reads animalId
reads animalFeedState
reads animalWalkState
reads animalHealthState
reads animalAdoptionState
reads taskStatus

Responsibilities:

Liveness:

```
ANIMAL_ASSISTANT = (
   (FeedRequest . ConfirmFeedTask) |
   (WalkRequest . ConfirmWalkTask) |
   (HealthRequest . ConfirmHealthRequestTask) |
   ConfirmInitialCheckupTask |
   ConfirmAdoptionApplicationTask
)*
```

Safety:

animalFeedState(animalId)="hungry" ∧ noneAssigned(feed,animalId) ⇒ newFeedRequest(animalId)
animalWalkState(animalId)="needsWalk" ∧ noneAssigned(walk,animalId) ⇒ newWalkRequest(animalId)
animalHealthState(animalId)="sick" ∧ noneAssigned(health,animalId) ⇒ newHealthRequest(animalId)

∀ taskId: ConfirmFeedTask(taskId,animalId) ⇒ concerns(taskId,animalId)

∀ taskId: ConfirmWalkTask(taskId,animalId) ⇒ concerns(taskId,animalId)

∀ taskId: ConfirmHealthRequestTask(taskId,animalId) ⇒ concerns(taskId,animalId)

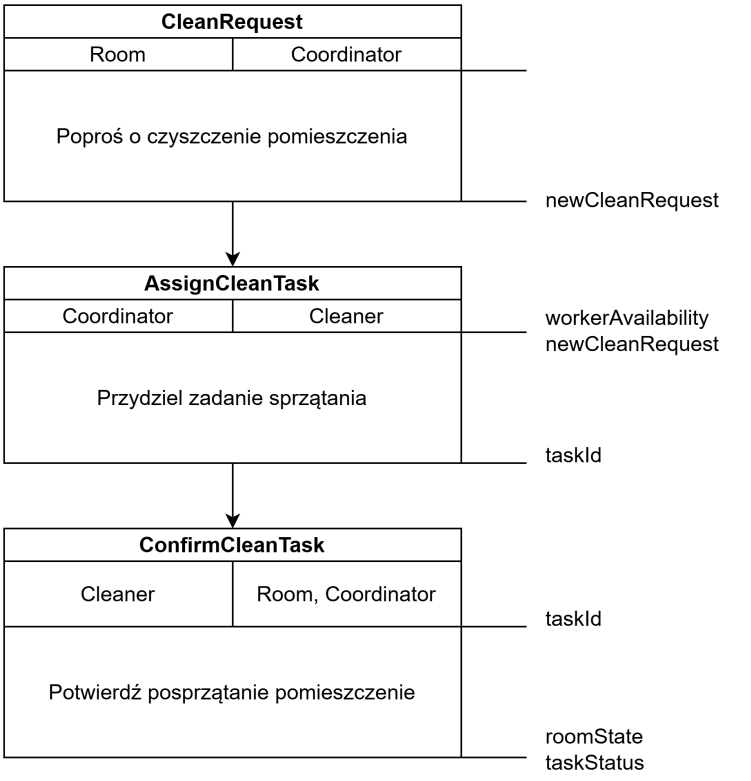
Role Schema: Pomieszczenie (Room)
Description: generuje zadania sprzątania.
Protocols and Activities: CleanRequest, ConfirmCleanTask
Permissions: generates newCleanRequest reads roomId reads roomState reads taskStatus
Responsibilities: Liveness: ROOM = (CleanRequest . ConfirmCleanTask)* Safety: roomState(roomId)="dirty" ∧ noneAssigned(clean,roomId) ⇒ newCleanRequest(roomId) ∀ taskId: ConfirmCleanTask(taskId,roomId) ⇒ concerns(taskId,roomId) (tylko zadanie dotyczące tego pomieszczenia).

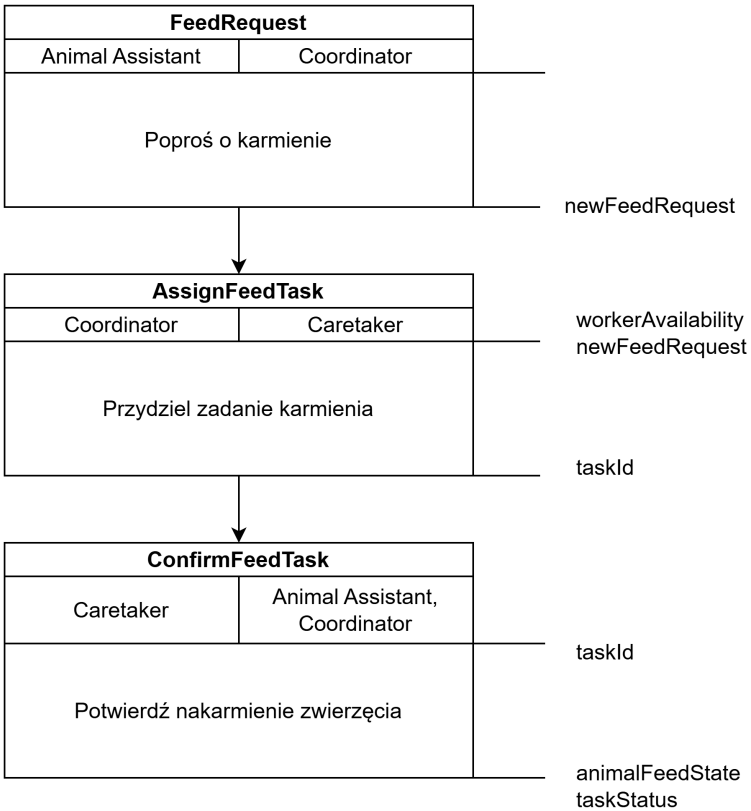
Role Schema: Recepcja (Reception)
Description: obsługuje proces przyjęcia zwierząt do schroniska, w tym rejestrację nowych podopiecznych, a następnie żądanie dodania ich do harmonogramu opieki.
Protocols and Activities: RegisterNewAnimal
Permissions: generates newAnimalRegistration
Responsibilities: Liveness: RECEPTION = RegisterNewAnimal* Safety: true

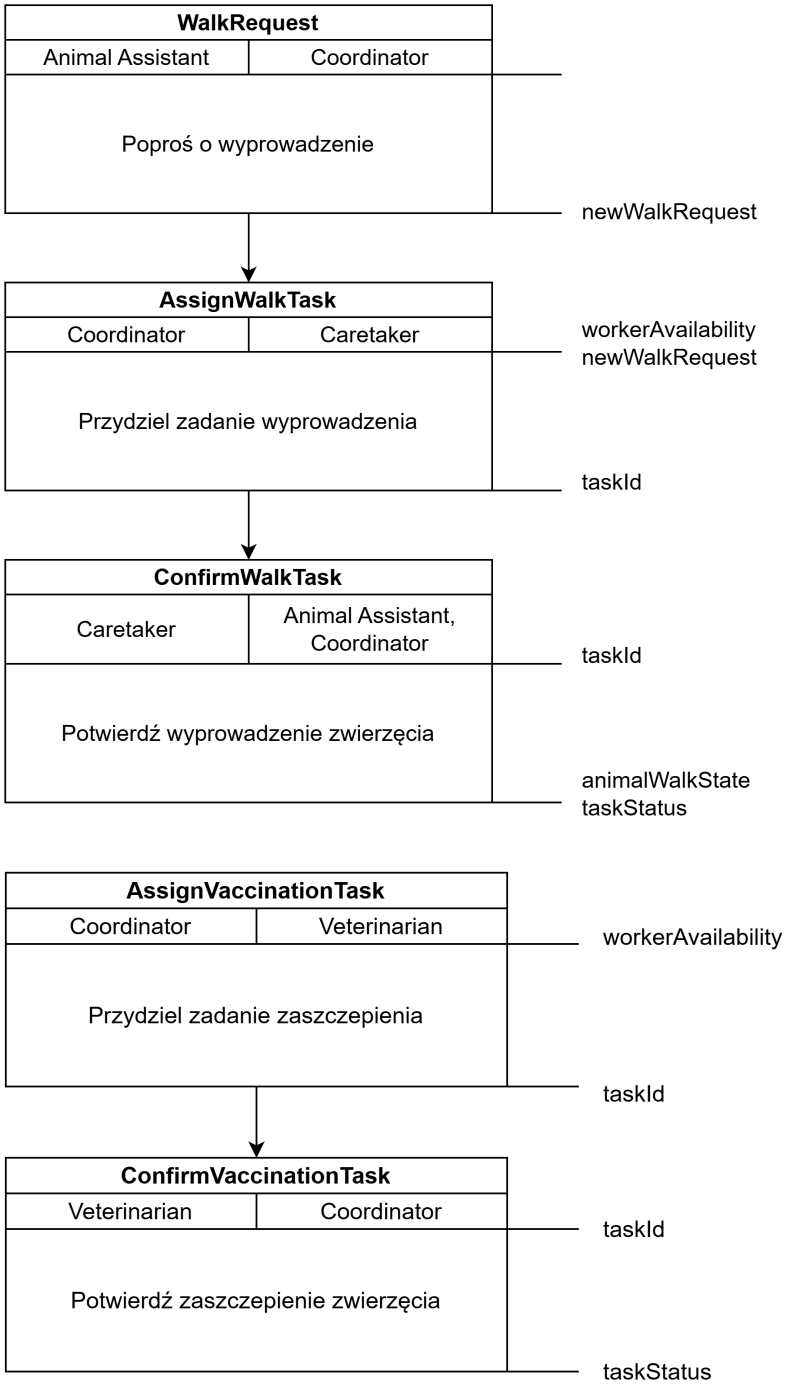
Role Schema: Adopcja (Adoption)
Description: prowadzi proces adopcyjny, obejmujący obsługę wniosków.
Protocols and Activities: AdoptionApplicationTask, ConfirmAdoptionApplicationTask
Permissions: generates animalAdoptionState generates taskStatus reads animalId reads taskId
Responsibilities: Liveness: ADOPTION = (AdoptionApplicationTask . ConfirmAdoptionApplicationTask)* Safety: ∀ taskId: ConfirmAdoptionApplicationTask(taskId,animalId) ⇒ set(animalAdoptionState(animalId),"adopted") ∧ set(taskStatus(taskId),"done")

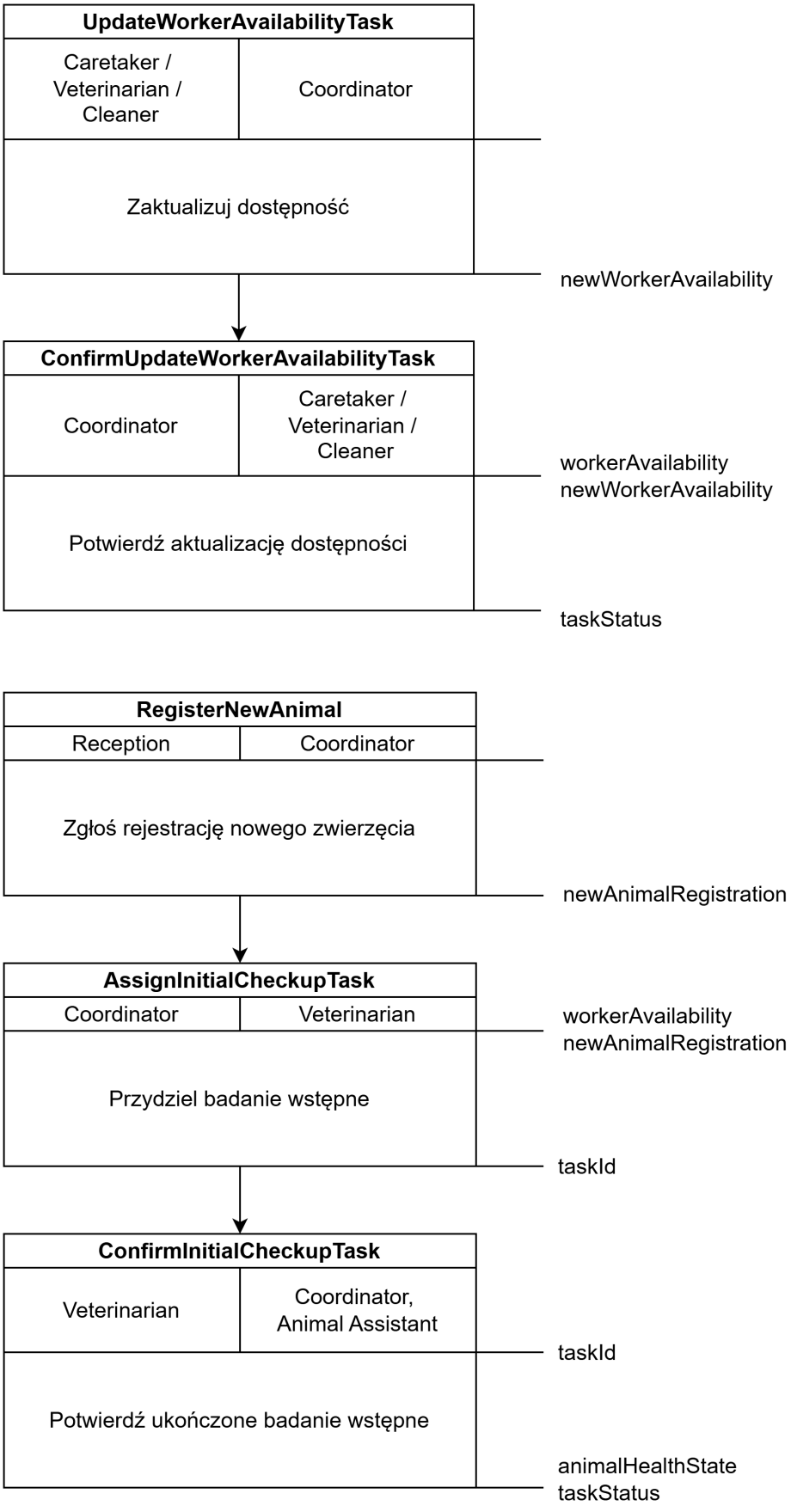
Role Schema: Wnioskodawca (Applicant)
Description: reprezentuje osobę ubiegającą się o adopcję.
Protocols and Activities: AdoptionApplicationRequest, ConfirmAdoptionApplicationTask
Permissions: generates newAdoptionApplication reads animalId reads animalAdoptionState reads taskStatus
Responsibilities: Liveness: APPLICANT = (AdoptionApplicationRequest . ConfirmAdoptionApplicationTask)* Safety: animalAdoptionState(animalId)="available" ∧ noneAssigned(adoption,animalId) ⇒ newAdoptionApplication(animalId) ∀ taskId: ConfirmAdoptionApplicationTask(taskId,animalId) ⇒ concerns(taskId,animalId) (potwierdza tylko swój wniosek)

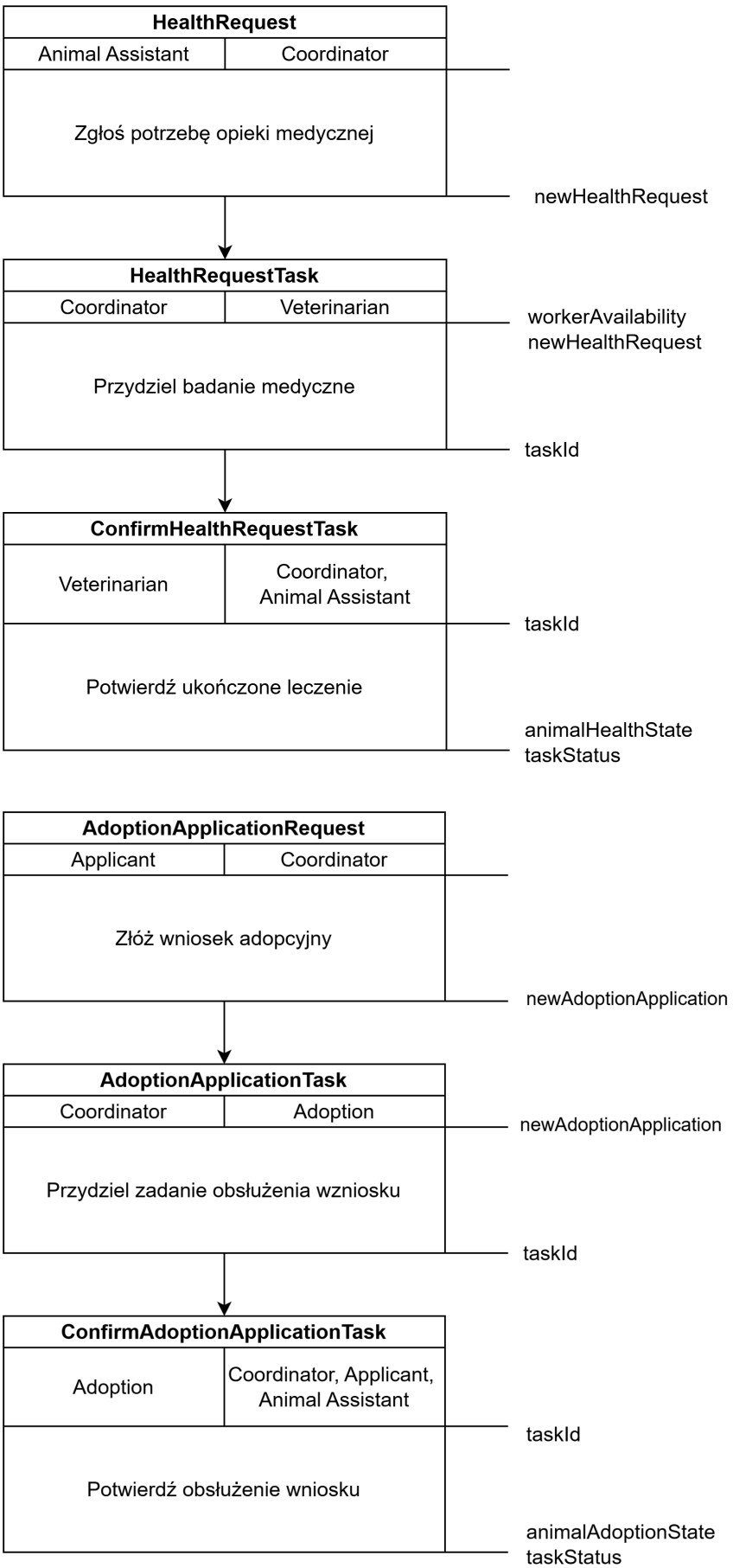
Model interakcji:



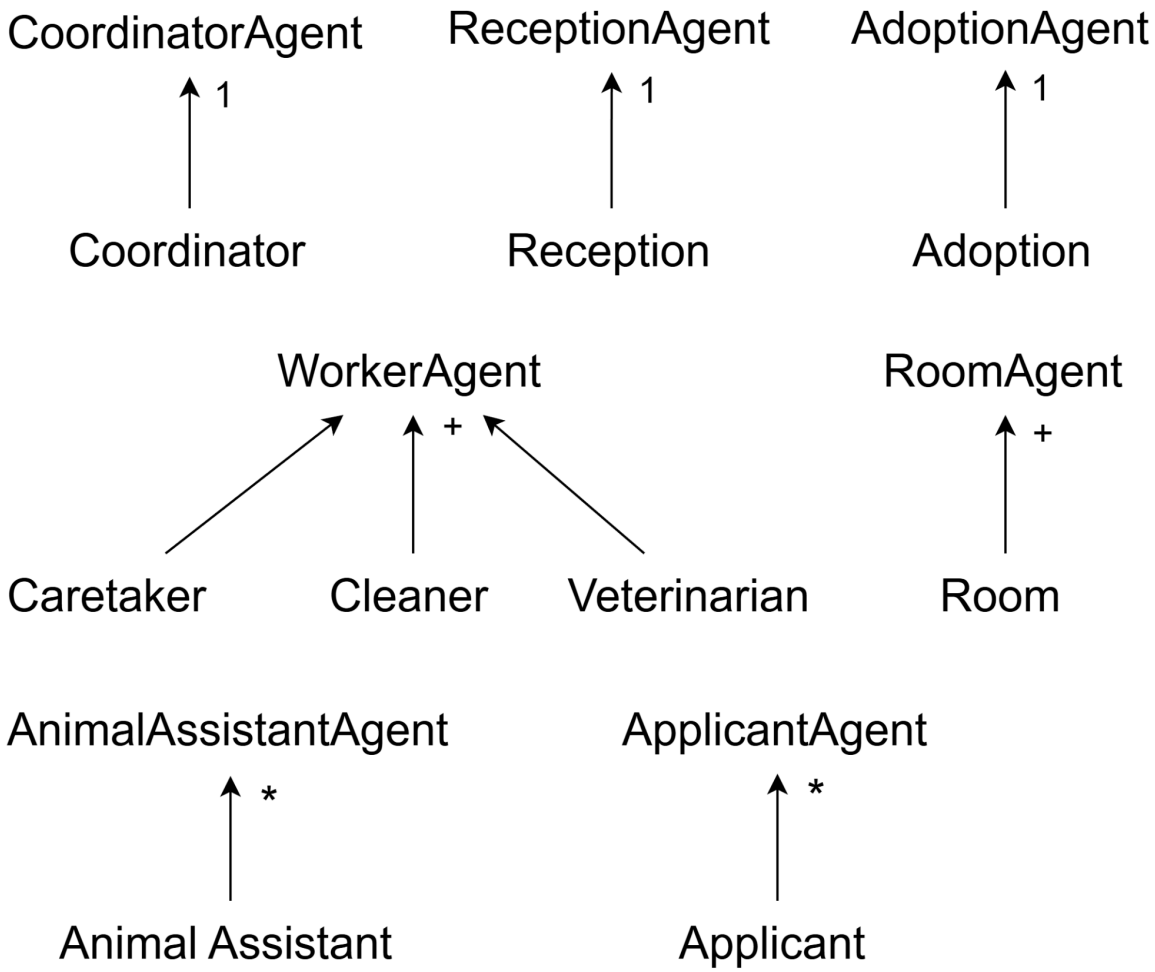



WalkRequest
Animal Assistant
Coordinator
Poproś o wyprowadzenie
newWalkRequest
AssignWalkTask
Coordinator
Caretaker
workerAvailability
newWalkRequest
Przydziel zadanie wyprowadzenia
taskId
ConfirmWalkTask
Caretaker
Animal Assistant, Coordinator
taskId
Potwierdź wyprowadzenie zwierzęcia
animalWalkState
taskStatus
AssignVaccinationTask
Coordinator
Veterinarian
workerAvailability
Przydziel zadanie zaszczepienia
taskId
ConfirmVaccinationTask
Veterinarian
Coordinator
taskId
Potwierdź zaszczepienie zwierzęcia
taskStatus


UpdateWorkerAvailabilityTask
Caretaker / Veterinarian / Cleaner
Coordinator
Zaktualizuj dostępność
newWorkerAvailability
ConfirmUpdateWorkerAvailabilityTask
Coordinator
Caretaker / Veterinarian / Cleaner
workerAvailability
newWorkerAvailability
Potwierdź aktualizację dostępności
taskStatus
RegisterNewAnimal
Reception
Coordinator
Zgłoś rejestrację nowego zwierzęcia
newAnimalRegistration
AssignInitialCheckupTask
Coordinator
Veterinarian
workerAvailability
newAnimalRegistration
Przydziel badanie wstępne
taskId
ConfirmInitialCheckupTask
Veterinarian
Coordinator, Animal Assistant
taskId
Potwierdź ukończone badanie wstępne
animalHealthState
taskStatus


HealthRequest
Animal Assistant
Coordinator
Zgłoś potrzebę opieki medycznej
newHealthRequest
HealthRequestTask
Coordinator
Veterinarian
workerAvailability
newHealthRequest
Przydziel badanie medyczne
taskId
ConfirmHealthRequestTask
Veterinarian
Coordinator,
Animal Assistant
taskId
Potwierdź ukończone leczenie
animalHealthState
taskStatus
AdoptionApplicationRequest
Applicant
Coordinator
Złóż wniosek adopcyjny
newAdoptionApplication
AdoptionApplicationTask
Coordinator
Adoption
newAdoptionApplication
Przydziel zadanie obsłużenia wzniosku
taskId
ConfirmAdoptionApplicationTask
Adoption
Coordinator, Applicant,
Animal Assistant
taskId
Potwierdź obsłużenie wniosku
animalAdoptionState
taskStatus

Model agentów:



Model usług:

Usługa	Wejścia	Wyjścia	Warunki wstępne	Warunki końcowe
request cleaning	roomId roomState	newCleanRequest	roomState="dirty" ∧ noneAssigned(clean, roomId)	newCleanRequest ≠ NULL
assign clean task	newCleanRequest workerAvailability	taskId taskStatus	newCleanRequest ≠ NULL	taskStatus="inProgress"
confirm clean task	taskId roomId taskStatus	roomState taskStatus	taskStatus="inProgress"	roomState="clean" ∧ taskStatus="done"
request feeding	animalId animalFeedState	newFeedRequest	animalFeedState="hungry" ∧ noneAssigned(feed,animalId)	newFeedRequest ≠ NULL
assign feed task	newFeedRequest workerAvailability	taskId taskStatus	newFeedRequest ≠ NULL	taskStatus="inProgress"
confirm feed task	taskId animalId taskStatus	animalFeedState taskStatus	taskStatus="inProgress"	animalFeedState="fed" ∧ taskStatus="done"
request walk	animalId animalWalkState	newWalkRequest	animalWalkState="needsWalk" ∧ noneAssigned(walk, animalId)	newWalkRequest ≠ NULL
assign walk task	newWalkRequest workerAvailability	taskId taskStatus	newWalkRequest ≠ NULL	taskStatus="inProgress"
confirm walk task	taskId animalId taskStatus	animalWalkState taskStatus	taskStatus="inProgress"	animalWalkState="walked" ∧ taskStatus="done"
request health care	animalId animalHealthState	newHealthRequest	animalHealthState=”sick” ∧ noneAssigned(health,animalId)	newHealthRequest ≠ NULL
assign health request task	newHealthRequest workerAvailability	taskId taskStatus	newHealthRequest ≠ NULL	taskStatus="inProgress"
confirm health request task	taskId animalId taskStatus	animalHealthState taskStatus	taskStatus="inProgress"	animalHealthState="healthy" ∧ taskStatus="done"
assign vaccination task	workerAvailability	taskId taskStatus	dueForVaccination(animalId)	taskStatus="inProgress"
confirm vaccination task	taskId taskStatus	taskStatus	taskStatus="inProgress"	taskStatus="done"
register new animal	animalPayload	newAnimalRegistration	true	newAnimalRegistration ≠ NULL

assign initial checkup	newAnimalRegistration workerAvailability	taskId taskStatus	newAnimalRegistration ≠ NULL	taskStatus="inProgress"
confirm initial checkup	taskId animalId taskStatus	animalHealthState taskStatus	taskStatus="inProgress"	animalHealthState="healthy" ∧ taskStatus="done"
submit adoption application	animalId applicantId animalAdoptionState	newAdoptionApplication	animalAdoptionState="available" ∧ noneAssigned(adoption,animalId)	newAdoptionApplication ≠ NULL
adoption application task	newAdoptionApplication	taskId taskStatus	newAdoptionApplication ≠ NULL	taskStatus="inProgress"
confirm adoption application	taskId animalId taskStatus	animalAdoptionState taskStatus	taskStatus="inProgress"	animalAdoptionState="adopted" ∧ taskStatus="done"
update availability	workerId	newWorkerAvailability	true	newWorkerAvailability ≠ NULL
confirm availability update	workerId newWorkerAvailability	taskStatus	newWorkerAvailability ≠ NULL	workerAvailability(workerId) = newWorkerAvailability

Model znajomości: