**NFTPet — Документация по функциям контракта**

**Создание и управление питомцами**

**mintPet(name: string, tokenURI: string)**

Создаёт нового питомца с заданным именем и метаданными (URI).  
**Входные данные:**

* name — имя питомца
* tokenURI — URI с метаданными (ссылка на JSON)  
  **Доступно:** всем  
  **Использование:** вызывается при первом создании питомца (через getPet).

**getPet(name: string, tokenURI: string) payable**

Основной метод для получения питомца:

* Если у пользователя нет питомцев, питомец создаётся бесплатно.
* Если есть — требуется оплата PET\_PRICE (0.01 ETH по умолчанию).  
  **Входные данные:**
* name — имя питомца
* tokenURI — URI метаданных

**метаданные питомцаTokenURI** хрянят всю информацию о питомце(имя, изображение, возраст, могу по желанию добавить описание и цену. При создании питомца всю введенную пользователем информацию записывать в отдедьный json файл, все их хранить в отдельной папке, залить на и при создании передавать ссылку на него. У каждого питомца отдельный файл.

**Оплата:** обязательно, если у пользователя уже есть питомец  
**События:** PetCreated с ID питомца и адресом владельца.

**sellPet(tokenId: uint256, to: address)**

Продаёт питомца другому пользователю, переводя NFT и обновляя владельцев.  
**Требования:**

* Вызывать может только текущий владелец питомца.
* Питомец должен быть живым (PetState.Active).  
  **Входные данные:**
* tokenId — ID питомца
* to — адрес нового владельца.

**Работа со здоровьем и опытом питомца**

**getHealth(tokenId: uint) view returns (uint)**

Возвращает текущее здоровье питомца с учётом времени с последнего кормления.  
**Входные данные:**

* tokenId — ID питомца  
  **Возвращает:** текущее здоровье (0-100).

**feedPet(tokenId: uint)**

Кормит питомца, восстанавливает здоровье, даёт опыт и может вызвать эволюцию.  
**Ограничения:**

* Можно кормить не чаще, чем раз в 5 минут.
* Питомец должен быть живым.  
  **Входные данные:**
* tokenId — ID питомца  
  **События:** PetFed, PetDied, EvolutionStage.

**feedPetBonus(tokenId: uint) payable**

"Бонусное" кормление за оплату, восстанавливает здоровье до максимума и даёт опыт.  
**Оплата:** минимум BONUS\_FEED\_PRICE (0.005 ETH).  
**Ограничения:** можно кормить не чаще, чем раз в час.  
**Входные данные:**

* tokenId — ID питомца  
  **События:** PetFed.  
  **Если оплата больше:** сдача возвращается отправителю.

**decayExperience(tokenId: uint)**

Уменьшает опыт питомца за пропущенные дни без кормления (5 единиц опыта в день).  
**Входные данные:**

* tokenId — ID питомца

**Обновление данных питомца**

**updateTokenURI(tokenId: uint, tokenURI: string)**

Обновляет метаданные питомца (URI) – после еволюционирования(повышения возраста +1 год , происходит при получении очков выше чем в массиве  uint[] public experienceThresholds = [0, 20, 70, 150, 250, 370, 500, 650, 730, 1000];

**Требования:** только владелец питомца может вызвать.  
**Входные данные:**

* tokenId — ID питомца
* tokenURI — новый URI.

**burnDeadPet(tokenId: uint)**

Удаляет из контракта мёртвого питомца и его NFT.  
**Требования:** питомец должен быть мёртвым.  
**Входные данные:**

* tokenId — ID питомца

**Получение информации о питомцах**

**getPetStatus(tokenId: uint) view returns (name, health, lastFed, age, state)**

Возвращает текущий статус питомца:

* имя
* здоровье
* время последнего кормления
* возраст (этап эволюции)
* состояние (жив или мёртв)

**getName(tokenId: uint) view returns (string)**

Получить имя питомца по ID.

**getPetState(tokenId: uint) view returns (PetState)**

Получить состояние питомца (Active или Dead).

**getAge(tokenId: uint) view returns (uint)**

Получить возраст (этап эволюции) питомца.

**exists(tokenId: uint) view returns (bool)**

Проверить, существует ли питомец с таким ID.

**getMyPets() view returns (uint256[])**

Получить список всех питомцев (ID), принадлежащих вызывающему адресу.

**Управление ценами и настройками (только владелец контракта)**

**setPET\_PRICE(\_price: uint)**

Установить цену покупки питомца.

**setBONUS\_FEED\_PRICE(\_price: uint)**

Установить цену бонусного кормления.

**addExperienceThreshold(newThreshold: uint)**

Добавить новый порог опыта для эволюции питомца.  
**Требования:** новый порог должен быть строго больше предыдущего.

**withdraw()**

Владелец контракта выводит все средства, хранящиеся на контракте.

**Дополнительная информация**

**getPET\_PRICE() view returns (uint)**

Получить текущую цену покупки питомца.

**getBONUS\_FEED\_PRICE() view returns (uint)**

Получить текущую цену бонусного кормления.

**getContractBalance() view returns (uint)**

Получить баланс эфира на контракте.

**События контракта**

* PetCreated(uint tokenId, address petOwner) — создан новый питомец
* PetFed(uint tokenId, uint newHealth) — питомец покормлен
* PetDied(uint tokenId) — питомец умер
* EvolutionStage(uint tokenId, uint age) — питомец эволюционировал (изменился возраст)