МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ ПО РАЗРАБОТКЕ СИСТЕМЫ УПРАВЛЕНИЯ АВТОЗАПРАВОЧНОЙ СТАНЦИЕЙ НА PYTHON

по дисциплине
«Информатика и программирование»

Студент
гр. БИН-25-2 _____С.А Головцов

Ассистент
преподавателя _____ М.В.
Водяницкий

Владивосток 2026

Задание

Техническое задание — Система управления автозаправочной станцией Вы рабо таете в российской компании «НефтеСофт», разрабатывающей программное обеспечение для автоматизации автозаправочных станций. Вам поручено создать консольный прототип системы управления АЗС, который используется операторами и техническим персоналом станции.

Система должна учитывать реальные процессы работы заправки: 1) продажи топлива,

2) контроль запасов,
3) обслуживание цистерн,
4) аварийные ситуации и ведение статистики.

1. Общая идея программы

Программа представляет собой консольную систему управления автозаправочной станцией, которая позволяет:

– обслуживать клиентов (касса);
– контролировать запасы топлива;
– управлять цистернами и колонками;
– оформлять пополнение топлива;
– вести историю операций и статистику;
– обрабатывать аварийные ситуации.

Программа работает в виде меню с выбором действий и функционирует в непре рывном цикле до выхода пользователя.

2. Топливо и цистерны

На заправке используются следующие виды топлива:

1) АИ-92,
2) АИ-95,
3) АИ-98,
4) ДТ.

Для одного типа топлива может существовать несколько подземных цистерн. Каждая цистерна имеет:

– тип топлива;
– максимальный объем;
– текущий уровень топлива;
– состояние (включена / отключена); минимальный допустимый уровень.

Если уровень топлива в цистерне падает ниже минимального порога:
– цистерна автоматически отключается;
– из неё нельзя отпускать топливо.

После пополнения цистерна не включается автоматически— включение произво дится вручную через меню.

3. Колонки

На заправке есть 8 колонок. Каждая колонка поддерживает несколько типов топлива, и каждый пистолет подключен к конкретной цистерне.

Схема подключения:
– АИ-95 №1 → колонки 1–4
– АИ-95 №2 → колонки 5–8
– АИ-92 → колонки 1–6
– АИ-98 → колонки 3–6
– ДТ → колонки 3–8 4.

4. Главное меню программы

Пользователь может выбрать одно из действий:
– Обслужить клиента (касса)
– Проверить состояние цистерн
– Оформить пополнение топлива
– Баланс и статистика
– История операций
– Перекачка топлива между цистернами

– Включение / отключение цистерн
– Состояние колонок
– EMERGENCY — аварийная ситуация
– Выход

5. Функциональные требования

Подробное описание всех функций (продажа, по полнение, перекачка, авария, статистика, история) приведено в разделе 2.

6. Хранение данных

Все данные сохраняются в файлах (рекомендуется JSON):

– состояние цистерн;
– схема колонок;
– баланс и статистика;
– история операций.

Содержание

Введение

Автоматизация процессов на автозаправочных станциях является важнейшей задачей современной логистики, ритейла и управления транспортной инфраструктурой. В условиях постоянно растущего числа автомобилей и увеличения интенсивности транспортных потоков эффективное управление АЗС становится критически значимым для обеспечения бесперебойного снабжения топливом, оптимизации логистических цепочек и повышения экономической эффективности работы топливно-энергетического комплекса.

Современные автозаправочные станции представляют собой сложные технологические комплексы, включающие не только оборудование для хранения и отпуска топлива, но и системы управления, контроля, безопасности и учета. Ручное управление такими системами становится неэффективным, подверженным человеческим ошибкам и неспособным обеспечить необходимый уровень оперативности реагирования на изменяющиеся условия работы.

Консольные системы управления, несмотря на кажущуюся простоту интерфейса, предоставляют мощный инструмент для автоматизации ключевых процессов на АЗС. Они позволяют эффективно контролировать запасы топлива в режиме реального времени, точно фиксировать все финансовые операции, оперативно реагировать на аварийные ситуации и технические неисправности, обеспечивать бесперебойную работу инфраструктуры даже при ограниченных вычислительных ресурсах и в условиях ненадежных каналов связи.

Важным преимуществом консольных систем является их высокая надежность и стабильность работы. Отсутствие графического интерфейса снижает требования к аппаратным ресурсам,

минимизирует вероятность сбоев, связанных с визуальными компонентами, и обеспечивает работу системы на устаревшем или специализированном оборудовании. Это особенно актуально для удаленных АЗС, где доступ к современным вычислительным ресурсам может быть ограничен.

Целью данной лабораторной работы является разработка полнофункционального прототипа системы управления автозаправочной станцией, полностью соответствующего техническому заданию и реальным потребностям эксплуатации АЗС.

Таким образом, данная работа представляет собой комплексное решение, сочетающее в себе образовательную ценность, практическую полезность и технологическую современность. Разрабатываемая система служит наглядной демонстрацией того, как средства программирования могут быть использованы для автоматизации реальных бизнес-процессов и повышения эффективности работы важнейшей инфраструктурной отрасли.

1 Принцип проектирования

При разработке системы управления автозаправочной станцией были сознательно заложены и последовательно реализованы фундаментальные принципы проектирования программного обеспечения, которые обеспечивают высокое качество, надежность и долгосрочную поддерживаемость системы. Эти принципы отражают современные подходы к разработке промышленного программного обеспечения и учитывают специфические требования автоматизации критически важной инфраструктуры.

- Надежность: все операции проверяются на корректность, некорректный ввод не приводит к аварийному завершению.
- Состоятельность: система сохраняет полное состояние между запусками.
- Безопасность: аварийный режим блокирует все операции и требует ручного восстановления.
- Модульность: код разделён на логические компоненты, каждый из которых от- вечает за свою зону ответственности.

Эти принципы в совокупности обеспечивают соответствие системы требованиям промышленного программного обеспечения: надежность в работе, легкость сопровождения, возможность расширения и адаптации к изменяющимся требованиям, а также высокую производительность разработки и отладки. Архитектурные решения, заложенные в систему, позволяют ей служить прочной основой для создания более сложных и функциональных систем автоматизации АЗС в будущем.

2 Разработка системы управления АЗС

2.1 Общая структура и константы программы

На данном этапе была реализована базовая структура программы, подключение необходимых библиотек и объявление ключевых констант, используемых во всей системе управления АЗС.

```
import json
import os
from datetime import datetime
from typing import Dict, List, Optional

# Константы
DATA_FILE = "azs_data.json"
PRICES = {
    "АИ-92": 57.50,
    "АИ-95": 58.30,
    "АИ-98": 64.20,
    "ДТ": 56.80
}
```

Листинг 2.1 – Подключение библиотек и объявление глобальных констант.

В данном листинге представлены основные модули и константы, необходимые для работы всей системы. Они обеспечивают хранение цен на топливо, работу с файлами и обработку времени.

2.2 Реализация класса управления цистернами

Класс Cistern реализует ключевую бизнес-логику хранения топлива, контроля минимального уровня, отпуска и пополнения топлива на автозаправочной станции.

```python
class Cistern:
    def __init__(self, fuel_type: str, max_volume: float, min_level_percent: float = 10.0):
        self.fuel_type = fuel_type
        self.max_volume = max_volume
        self.current_volume = max_volume * 0.5  # Начинаем с 50% заполнения
        self.min_level = max_volume * (min_level_percent / 100)
        self.is_enabled = True
        self.id = f"{fuel_type}_{id(self)}"[-6:]

    def to_dict(self):
        return {
            "fuel_type": self.fuel_type,
            "max_volume": self.max_volume,
            "current_volume": self.current_volume,
            "min_level": self.min_level,
            "is_enabled": self.is_enabled,
            "id": self.id
        }

    @classmethod
    def from_dict(cls, data):
        cistern = cls(data["fuel_type"], data["max_volume"])
        cistern.current_volume = data["current_volume"]
        cistern.min_level = data["min_level"]
        cistern.is_enabled = data["is_enabled"]
        cistern.id = data["id"]
        return cistern

    def can_dispense(self, amount: float) -> bool:
        return self.is_enabled and self.current_volume >= amount

    def dispense(self, amount: float) -> bool:
        if self.can_dispense(amount):
            self.current_volume -= amount
            # Автоматическое отключение при низком уровне
            if self.current_volume < self.min_level:
                self.is_enabled = False
            return True
        return False

    def refill(self, amount: float) -> bool:
        if self.current_volume + amount <= self.max_volume:
            self.current_volume += amount
            return True
        return False
```

Листинг 2.2 – Основные методы класса управления цистернами.

Данный листинг демонстрирует логику хранения топлива, проверки доступности отпуска и автоматического отключения цистерны при снижении уровня топлива ниже допустимого.

2.3 Реализация класса топливных колонок

Класс FuelColumn отвечает за взаимодействие колонок с цистернами и организацию отпуска топлива клиентам.

```
70  class FuelColumn:
71      def __init__(self, column_id: int):
72          self.column_id = column_id
73          # Сопоставление типа топлива с цистерной
74          self.fuel_connections = {}
75
76      def connect_fuel(self, fuel_type: str, cistern: Cistern):
77          self.fuel_connections[fuel_type] = cistern
78
79      def get_available_fuels(self):
80          available = []
81          for fuel_type, cistern in self.fuel_connections.items():
82              if cistern.is_enabled:
83                  available.append((fuel_type, cistern))
84          return available
85
86      def dispense_fuel(self, fuel_type: str, amount: float) -> Optional[float]:
87          if fuel_type in self.fuel_connections:
88              cistern = self.fuel_connections[fuel_type]
89              if cistern.dispense(amount):
90                  return amount * PRICES.get(fuel_type, 0)
91          return None
```

Листинг 2.3 – Класс топливной колонки и отпуск топлива.

В листинге показан механизм подключения топлива к колонке, проверки доступности и расчета стоимости отпуска топлива.

2.4 Основной класс системы AZSSystem

Основной класс AZSSystem объединяет все элементы системы: цистерны, колонки, статистику и операции станции.

```
93  class AZSSystem:
94      def __init__(self):
95          self.cisterns = []
96          self.columns = []
97          self.transactions = []
98          self.stats = {
99              "total_cars": 0,
100             "total_income": 0.0,
101             "fuel_sold": {fuel: 0.0 for fuel in PRICES},
102             "fuel_income": {fuel: 0.0 for fuel in PRICES},
103             "refills": {fuel: 0.0 for fuel in PRICES},
104             "emergency_mode": False,
105             "emergency_start_time": None
106         }
107         self.initialize_system()
108         self.load_data()
```

Листинг 2.4 – Инициализация основной системы управления АЗС.

Данный фрагмент отражает архитектуру всей программы и создание ключевых структур данных, необходимых для функционирования автозаправочной станции.

2.5 Логика обслуживания клиентов

Метод обслуживания клиентов реализует основной алгоритм работы кассы: выбор колонки, топлива, ввод количества литров и расчет стоимости заправки.

```
272     def serve_customer(self):
273         if self.stats["emergency_mode"]:
274             print("\n!!! АВАРИЙНЫЙ РЕЖИМ !!!")
275             print("Заправка приостановлена. Возобновление работы невозможно до отключения аварийного режима.")
276             return
277 
278         print("\n--- Обслуживание клиента ---")
279 
280         # Показываем доступные колонки
281         print("\nДоступные колонки:")
282         for i, column in enumerate(self.columns, 1):
283             available_fuels = column.get_available_fuels()
284             if available_fuels:
285                 print(f"{i}) Колонка {i}")
286 
287         try:
288             col_choice = int(input("\nВыберите колонку: ")) - 1
289             if not (0 <= col_choice < len(self.columns)):
290                 print("Неверный номер колонки.")
291                 return
292 
293             column = self.columns[col_choice]
294             available_fuels = column.get_available_fuels()
295 
296             if not available_fuels:
297                 print("На этой колонке нет доступного топлива.")
298                 return
299 
300             print(f"\nКолонка {col_choice + 1}")
301             print("\nДоступные виды топлива:")
302             for i, (fuel_type, cistern) in enumerate(available_fuels, 1):
303                 print(f"{i}) {fuel_type} (цистерна {cistern.id})")
304 
305             fuel_choice = int(input("\nВыберите тип топлива: ")) - 1
306             if not (0 <= fuel_choice < len(available_fuels)):
307                 print("Неверный выбор топлива.")
308                 return
309 
310             fuel_type, cistern = available_fuels[fuel_choice]
```

Листинг 2.5 – Алгоритм обслуживания клиента на АЗС.

В листинге реализован ключевой бизнес-процесс станции — обслуживание клиента, включая выбор топлива, ввод объема и расчет стоимости заправки.

2.6 Главное меню и запуск программы

Главное меню обеспечивает управление всеми функциями системы и запуск программы.

```
734  def main():
735      system = AZSSystem()
736      system.show_main_menu()
737
738  if __name__ == "__main__":
739      main()
```

Листинг 2.6 – Главное меню и точка входа в программу.

Данный листинг завершает структуру приложения и обеспечивает запуск всей системы управления АЗС через главное пользовательское меню.

3. Заключение

В ходе выполнения работы была разработана полноценная система управления автозаправочной станцией на языке Python. В процессе разработки были реализованы ключевые модули учета, управления и обработки данных. Программный код структурирован, разбит на логические части и соответствует требованиям задания. Поставленные цели разработки достигнуты в полном объеме.