Приложение

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный аграрный университет имени П.А. Столыпина»

Университетский колледж агробизнеса

09.02.11 Разработка и управление программным обеспечением

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по ПП.02.01 Производственная практика (по профилю специальности)

Обеспечивающее преподавание дисциплины подразделение	Инженерное отделение
Разработчик:	
Преподаватель	А.В. Кортусов

Омск
2026

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ

1. Фонд оценочных средств (далее – ФОС) предназначен для контроля и оценки образовательных достижений обучающихся, освоивших программу ПП.02.01 Производственная практика (по профилю специальности).
2. ФОС включает оценочные материалы для проведения текущего контроля и промежуточной аттестации в форме зачета.
3. ФОС позволяет оценивать знания, умения, направленные на формирование компетенций.
4. ФОС разработан на основании положений основной образовательной программы по специальности 09.02.11 Разработка и управление программным обеспечением и рабочей программы ПП.02.01 Производственная практика (по профилю специальности).
5. ФОС является обязательным обособленным приложением к рабочей программе.

2. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ ИЗУЧЕНИЯ

Результаты обучения (освоенные умения, навыки)	**Показатели оценки образовательных результатов**
анализировать требования к программному обеспечению и составлять планы тестирования	Обучающийся умеет анализировать требования к программному обеспечению и составлять планы тестирования
выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестировании, анализировать результаты тестирования и документировать найденные ошибки	Обучающийся умеет выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестировании, анализировать результаты тестирования и документировать найденные ошибки
разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении	Обучающийся умеет разрабатывать стратегии гладки и исправлять ошибки в программном беспечении
описывать функциональность модулей в документации, создавать диаграммы для иллюстрации работы модулей	Обучающийся умеет описывать функциональность модулей в документации, создавать диаграммы для иллюстрации работы модулей
разбивать модули на логические блоки и описывать каждый блок отдельно включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки	Обучающийся умеет разбивать модули на логические блоки и описывать каждый блок отдельно включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки
вести журнал изменений и фиксировать обновления программных модулей, проводить регулярное обновление документации при изменении модулей или добавлении нового функционала	Обучающийся умеет вести журнал изменений и фиксировать обновления программных модулей, проводить регулярное обновление документации при изменении модулей или добавлении нового функционала
отладки программного обеспечения на уровне программных модулей	Обучающийся владеет навыками отладки программного обеспечения на уровне программных модулей
тестирования программного обеспечения и формирования тестовых сценариев подготовки тестовых платформ (установка операционной системы, дополнительного ПО и другого по необходимости)	Обучающийся владеет навыками тестирования программного обеспечения и формирования тестовых сценариев подготовки тестовых платформ (установка операционной системы, дополнительного ПО и другого по необходимости)
настройки тестовой среды и аппаратных средств для выполнения тестирования ПО в соответствии с заданием на тестирование в пределах своей компетенции	Обучающийся владеет навыками настройки тестовой среды и аппаратных средств для выполнения тестирования ПО в соответствии с заданием на тестирование в пределах своей компетенции
формирования и представления отчетности о подготовке к выполнению задания на тестирование ПО в соответствии с документирования установленными регламентами	Обучающийся владеет навыками формирования и представления отчетности о подготовке к выполнению задания на тестирование ПО в соответствии с документирования установленными регламентами
создания технической документации для модулей	Обучающийся владеет навыками создания технической документации для модулей

кода, API и интерфейсов, работы со специализированным ПО по документированию программного кода	Обучающийся владеет навыками кода, API и интерфейсов, работы со специализированным ПО по документированию программного кода

3. РАСПРЕДЕЛЕНИЕ ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ И ТИПОВ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ЭЛЕМЕНТАМ УМЕНИЙ И НАВЫКОВ

Содержание курса	Форма контроля	Умения	Навыки
Текущий контроль			
Раздел 1. Организационный этап			
Прохождение вводного инструктажа. Получение и обсуждение задания на практику	Проверка отчета по учебной практике	Уо 01.01, Уо 01.02, Уо 09.01	-
Раздел 2. Основной этап			
Выполнение работ:	наблюдение и оценка в процессе практики; анализ отчетной документации; экспертная оценка выполнения индивидуальных заданий.	У 2.4.1, У 2.4.2, У 2.4.3, У 2.5.1, У 2.5.2, У 2.5.3, Уо 01.01, Уо 01.02, Уо 09.01	Н 2.4.1, Н 2.4.2, Н 2.4.3, Н 2.4.4, Н 2.5.1, Н 2.5.2
Раздел 3. Заключительный этап			
Оформление введения. Оформление основной части. Оформление заключения. Оформление списка использованных источников и приложений. Оформление отчета и приложений. Прохождение собеседования (зачет)	Проверка отчета по учебной практике, собеседование	Уо 01.01, Уо 01.02, Уо 09.01	-
Промежуточный контроль			
Зачет	Проверка отчета по учебной практике, собеседование	У 2.4.1, У 2.4.2, У 2.4.3, У 2.5.1, У 2.5.2, У 2.5.3, Уо 01.01, Уо 01.02, Уо 09.01	Н 2.4.1, Н 2.4.2, Н 2.4.3, Н 2.4.4, Н 2.5.1, Н 2.5.2

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОЦЕНКИ УМЕНИЙ, НАВЫКОВ

4.1. Оценочные средства, применяемые для текущего контроля.

4.1.1. Разработка алгоритмов и программных модулей

Разработайте программный модуль для учета сотрудников предприятия на языке Python:

1. Создайте класс Employee с полями: id, full_name, birth_date, position, salary, department.
2. Реализуйте функции для выполнения CRUD-операций с использованием SQLite.
3. Разработайте модуль валидации данных (ФИО - минимум 2 слова, дата рождения - возраст от 16 до 65 лет, оклад - положительное число).
4. Реализуйте функцию поиска сотрудников по ФИО, должности и диапазону оклада.

Пример кода:

```
# models.py
from dataclasses import dataclass
from typing import Optional, List
import sqlite3
from datetime import datetime

@dataclass
class Employee:
    id: Optional[int] = None
    full_name: str = ""
    birth_date: str = ""
    position: str = ""
    salary: float = 0.0
    department: str = ""

class EmployeeValidator:
    @staticmethod
    def validate_full_name(name: str) -> bool:
        return len(name.strip().split()) >= 2

    @staticmethod
    def validate_birth_date(date: str) -> bool:
        try:
            age = (datetime.now() - datetime.strptime(date, '%Y-%m-%d')).days // 365
            return 16 <= age <= 65
        except:
            return False

    @staticmethod
    def validate_salary(salary: float) -> bool:
        return salary > 0

class EmployeeRepository:
    def __init__(self, db_path='employees.db'):
        self.db_path = db_path
        self._init_db()
```

```
    def _init_db(self):
        with sqlite3.connect(self.db_path) as conn:
            conn.execute('''
                CREATE TABLE IF NOT EXISTS employees (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    full_name TEXT NOT NULL,
                    birth_date TEXT NOT NULL,
                    position TEXT NOT NULL,
                    salary REAL NOT NULL,
                    department TEXT
                )
            ''')

    def add_employee(self, emp: Employee) -> Optional[int]:
        if not all([
            EmployeeValidator.validate_full_name(emp.full_name),
            EmployeeValidator.validate_birth_date(emp.birth_date),
            EmployeeValidator.validate_salary(emp.salary)
        ]):
            return None

        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.execute('''
                INSERT INTO employees (full_name, birth_date, position, salary, department)
                VALUES (?, ?, ?, ?, ?)
            ''', (emp.full_name, emp.birth_date, emp.position, emp.salary, emp.department))
            return cursor.lastrowid

    def get_employee(self, emp_id: int) -> Optional[Employee]:
        with sqlite3.connect(self.db_path) as conn:
            row = conn.execute('SELECT * FROM employees WHERE id = ?',
(emp_id,)).fetchone()
            return Employee(*row) if row else None

    def update_employee(self, emp: Employee) -> bool:
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.execute('''
                UPDATE employees SET full_name=?, birth_date=?, position=?, salary=?,
department=?
                WHERE id=?
            ''', (emp.full_name, emp.birth_date, emp.position, emp.salary, emp.department,
emp.id))
            return cursor.rowcount > 0

    def delete_employee(self, emp_id: int) -> bool:
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.execute('DELETE FROM employees WHERE id = ?', (emp_id,))
            return cursor.rowcount > 0
```

```
    def search_employees(self, full_name=None, position=None, salary_from=None, salary_to=None) -> List[Employee]:
        conditions, params = [], []
        if full_name:
            conditions.append("full_name LIKE ?")
            params.append(f"%{full_name}%")
        if position:
            conditions.append("position = ?")
            params.append(position)
        if salary_from:
            conditions.append("salary >= ?")
            params.append(salary_from)
        if salary_to:
            conditions.append("salary <= ?")
            params.append(salary_to)

        where = f"WHERE {' AND '.join(conditions)}" if conditions else ""
        with sqlite3.connect(self.db_path) as conn:
            rows = conn.execute(f'SELECT * FROM employees {where} ORDER BY full_name', params).fetchall()
            return [Employee(*row) for row in rows]
```

Форма выполнения: Исходный код модуля, примеры вызова функций, скриншоты работы.

4.1.2. Отладка и тестирование программного модуля

Выполните отладку и тестирование разработанного модуля с использованием PyTest:

```
python
# test_module.py
import pytest
import tempfile
import os
from models import Employee, EmployeeRepository, EmployeeValidator

class TestEmployeeValidator:
    def test_valid_full_name(self):
        assert EmployeeValidator.validate_full_name("Иванов Иван") is True
        assert EmployeeValidator.validate_full_name("Иванов") is False

    def test_valid_birth_date(self):
        assert EmployeeValidator.validate_birth_date("1990-01-01") is True
        assert EmployeeValidator.validate_birth_date("2010-01-01") is False

    def test_valid_salary(self):
        assert EmployeeValidator.validate_salary(50000) is True
        assert EmployeeValidator.validate_salary(-1000) is False

class TestEmployeeRepository:
    @pytest.fixture
    def repo(self):
        fd, path = tempfile.mkstemp(suffix='.db')
        os.close(fd)
```

```
    yield EmployeeRepository(path)
    os.unlink(path)

def test_add_employee(self, repo):
    emp = Employee(full_name="Иванов Иван", birth_date="1990-01-01",
                   position="Программист", salary=50000)
    emp_id = repo.add_employee(emp)
    assert emp_id is not None
    assert repo.get_employee(emp_id).full_name == "Иванов Иван"

def test_add_invalid_employee(self, repo):
    emp = Employee(full_name="Иван", birth_date="2010-01-01",
                   position="", salary=-1000)
    assert repo.add_employee(emp) is None

def test_search_employees(self, repo):
    emp1 = Employee(full_name="Иванов Иван", birth_date="1990-01-01",
                    position="Программист", salary=50000)
    emp2 = Employee(full_name="Петров Петр", birth_date="1985-06-15",
                    position="Программист", salary=60000)
    repo.add_employee(emp1)
    repo.add_employee(emp2)

    results = repo.search_employees(position="Программист")
    assert len(results) == 2
    results = repo.search_employees(full_name="Иванов")
    assert len(results) == 1
```

Форма выполнения: Код тестов, отчет о результатах тестирования с описанием выявленных ошибок.

4.1.3. Оптимизация и рефакторинг кода

Выполните оптимизацию и рефакторинг программного кода:

До рефакторинга:

python

```
def process_employees():
    conn = sqlite3.connect('employees.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM employees')
    employees = cursor.fetchall()
    conn.close()

    for emp in employees:
        if emp[4] < 30000:
            bonus = emp[4] * 0.1
        elif emp[4] >= 30000 and emp[4] < 50000:
            bonus = emp[4] * 0.15
        elif emp[4] >= 50000 and emp[4] < 80000:
            bonus = emp[4] * 0.2
        else:
            bonus = emp[4] * 0.25
```

```
        conn = sqlite3.connect('employees.db')
        cursor = conn.cursor()
        cursor.execute('UPDATE employees SET bonus = ? WHERE id = ?', (bonus, emp[0]))
        conn.commit()
        conn.close()
```
После рефакторинга:
```
python
class BonusCalculator:
    @staticmethod
    def calculate(salary: float) -> float:
        if salary < 30000: return salary * 0.1
        if salary < 50000: return salary * 0.15
        if salary < 80000: return salary * 0.2
        return salary * 0.25

def process_employees():
    with sqlite3.connect('employees.db') as conn:
        employees = conn.execute('SELECT * FROM employees').fetchall()
        for emp in employees:
            bonus = BonusCalculator.calculate(emp[4])
            conn.execute('UPDATE employees SET bonus = ? WHERE id = ?', (bonus, emp[0]))
        conn.commit()
```
Форма выполнения: Код до и после рефакторинга с пояснением выполненных изменений.

4.1.4. Интеграция модулей в приложение

Объедините все модули в единое приложение:

```
# main.py
import logging
from models import Employee, EmployeeRepository

logging.basicConfig(level=logging.INFO)

class EmployeeApp:
    def __init__(self, db_path='employees.db'):
        self.repo = EmployeeRepository(db_path)

    def run(self):
        while True:
            print("\n===== Управление сотрудниками =====")
            print("1. Добавить сотрудника")
            print("2. Поиск сотрудников")
            print("3. Показать всех сотрудников")
            print("4. Удалить сотрудника")
            print("0. Выход")

            choice = input("Выберите действие: ").strip()

            if choice == '1':
                emp = Employee(
                    full_name=input("ФИО: ").strip(),
```

```
                birth_date=input("Дата рождения (ГГГГ-ММ-ДД): ").strip(),
                position=input("Должность: ").strip(),
                salary=float(input("Оклад: ").strip()),
                department=input("Отдел: ").strip()
            )
            emp_id = self.repo.add_employee(emp)
            print(f"Сотрудник добавлен с ID={emp_id}" if emp_id else "Ошибка добавления")

        elif choice == '2':
            results = self.repo.search_employees(
                full_name=input("ФИО (Enter для пропуска): ").strip() or None,
                position=input("Должность (Enter для пропуска): ").strip() or None
            )
            print(f"\nНайдено {len(results)} сотрудников:")
            for emp in results:
                print(f"{emp.id}: {emp.full_name} - {emp.position} - {emp.salary}")

        elif choice == '3':
            results = self.repo.search_employees()
            print(f"\nВсего {len(results)} сотрудников:")
            for emp in results:
                print(f"{emp.id}: {emp.full_name} - {emp.position} - {emp.salary} руб.")

        elif choice == '4':
            emp_id = int(input("Введите ID сотрудника: "))
            print("Сотрудник удален" if self.repo.delete_employee(emp_id) else "Сотрудник не найден")

        elif choice == '0':
            print("До свидания!")
            break

if __name__ == '__main__':
    EmployeeApp().run()
```

Форма выполнения: Полностью работающее приложение, скриншоты работы всех функций.

4.2. Оценочные средства, применяемые для промежуточной аттестации по итогам изучения дисциплины

Зачет проводится по завершении производственной практики.

Промежуточная аттестация в форме зачета осуществляется по результатам сдачи отчета по практике и с учетом текущего контроля успеваемости при выполнении всех видов текущего контроля.

Обучающиеся, не выполнившие виды работ, предусмотренные рабочей программой; пропустившие более 50% практики без уважительной причины, не допускаются к зачету.

Промежуточная аттестация таких лиц проводится только после прохождения ими всех видов текущего контроля.

V. ПОКАЗАТЕЛИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Уровень сформированности компетенций	**Оценка**	**Критерии оценивания по видам работ**	
		тестирование (процент правильных ответов)	**прочие виды работ по практике**
Высокий	Отлично	90-100%	Обучающийся глубоко и прочно усвоил теоретический и освоил практический материал. Дает логичные и грамотные ответы. Демонстрирует знание не только основного, но и дополнительного материала, быстро ориентируется, отвечая на дополнительные вопросы. Свободно справляется с поставленными задачами, аргументировано и верно обосновывает принятые решения.
Повышенный	Хорошо	70-89%	Обучающийся твердо знает программный материал, грамотно и по существу излагает его. Не допускает существенных неточностей при ответах на вопросы, правильно применяет теоретические положения при решении практических задач, владеет навыками и приемами их выполнения.
Базовый	Удовлетворительно	50-69%	Обучающийся демонстрирует знания только основного материала, но не усвоил его детали, испытывает затруднения при решении практических задач. В ответах на поставленные вопросы допускает неточности. Дает определения понятий, неискажающие их смысл. Нарушает последовательность изложения программного материала.
Не сформирована	Неудовлетворительно	0-49%	Обучающийся не знает, не выполняет или неправильно выполняет большую часть учебного материала. Допускает ошибки в формулировке определений, искажающие их смысл, беспорядочно и неуверенно излагает материал. Ответы на дополнительные вопросы отсутствуют. Не выполняет задания.

ЛИСТ РАССМОТРЕНИЙ И ОДОБРЕНИЙ

рабочей программы

ПП 02.01 Производственная практика (по профилю специальности)

в составе **ООП 09.02.11 Разработка и управление программным обеспечением**

1)Рассмотрена и одобрена:
а)На заседании предметно-цикловой методической комиссии протокол № 6 от 10.03.2026 г. Председатель ПЦМК _____ С.М. Нурбаева
б) На заседании методического совета протокол №4 от 21.04.2026 г. Председатель методического совета _____ М.В. Иваницкая
2)Рассмотрена и одобрена внешним экспертом
а) директор ООО «Сатори Партнер» А.Б. Мальцев