Е. В. ХУДЯКОВА
М. Н. СТЕПАНЦЕВИЧ
М. А. КАЧАЛИН
М. И. ГОРБАЧЕВ

ЦИФРОВЫЕ ПЛАТФОРМЫ В АПК

Учебно-методическое пособие

Рекомендовано УМО Института экономики и управления имени А. В. Чаянова ФГБОУ ВО РГАУ–МСХА имени К. А. Тимирязева

Москва
«Мегаполис»
2023

УДК 338.436.33+004.4
ББК 32.973.4+65.32
Ц 75

Рецензенты:
доктор экономических наук, профессор, заведующий кафедрой информационных систем ФГБОУ ВО «Кубанский ГАУ» **Е. В. Попова**

директор направления отраслевых сервисов для АПК ПАО «Ростелеком»
Д. В. Жуковский

Худякова Е. В., Степанцевич М. Н., Качалин М. А., Горбачев М. И.
Ц 75 Цифровые платформы в АПК: учебно-методическое пособие / Е. В. Худякова, М. Н. Степанцевич, М. А. Качалин, М. И. Горбачев / ФГБОУ ВО РГАУ−МСХА имени К. А. Тимирязева. – М. : ООО «Мегаполис», 2023. – 96 с.

ISBN 978-5-6049928-1-4

В первом разделе пособия излагаются характеристики цифровых платформ в АПК. Второй раздел учебно-методического пособия посвящен раскрытию особенностей реализации технологий распределенных реестров для интеграции элементов цифровой платформы, а также приведен пример разработки смарт-контракта на блокчейн-платформе.

Учебно-методическое пособие может быть рекомендовано студентам, обучающимся по направлениям «Прикладная информатика», «Информационные системы и технологии», «Экономика», «Агрономия» и другим.

Учебно-методическое пособие написано с использованием опыта работы, накопленного авторами в процессе преподавания учебных дисциплин, связанных с вопросами применения цифровых решений и технологий в АПК, подготовки и издания учебно-методических материалов в высших учебных заведениях, где они осуществляли свою научную и педагогическую деятельность. Изучение данного пособия требует знания основ экономики, математической статистики, теории вероятностей, информатики и программирования.

Учебно-методическое пособие рекомендуется к изданию учебно-методической комиссией Института экономики и управления имени А. В. Чаянова ФГБОУ ВО РГАУ–МСХА имени К.А. Тимирязева (протокол №2 от 07 октября 2022 года).

УДК 338.436.33+004.4
ББК 32.973.4+65.32

ISBN 978-5-6049928-1-4

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Агропромышленный комплекс является ведущей отраслью экономики нашей страны. По предварительным данным Федеральной службы государственной статистики, индекс производства продукции сельского хозяйства (в сопоставимых ценах) в хозяйствах всех категорий в 2021 году составил 105,7 % к 2019 году.

По оценке Министерства сельского хозяйства Российской Федерации, в 2021 году достигнуты или превышены пороговые значения уровня самообеспечения страны в отношении зерна, сахара, растительного масла, мяса и мясопродуктов, предусмотренные Доктриной продовольственной безопасности Российской Федерации. При этом в 2021 году объем экспорта продукции АПК составил 32 млрд долл., или на 6,2 % больше уровня 2019 года.

Однако, в силу произошедших изменений геополитического и экономического характера агропромышленный комплекс наравне с другими секторами экономики нашей страны столкнулся с многочисленными вызовами и угрозами, такими как:

- диспаритет ценовых пропорций на продукцию АПК и потребляемые в отрасли ресурсы, вследствие ориентации на внутреннем рынке на ценовые индикаторы, определяемые внешнеэкономической конъюнктурой;
- низкий уровень внутреннего платежеспособного спроса на продукцию АПК, в том числе высокомаржинальную («зеленую» и органическую) и, как следствие, существенный экспорт продукции с низким уровнем передела;
- сильное техническое и технологическое отставание (рассло-

ение) товаропроизводителей по отрасли в зависимости от уровня концентрации производства;

- стойкая зависимость отечественных аграриев от импортных семян, средств защиты, племенного материала, отставание в локализации критически важных видов сырья, ресурсов и др.;
- снижение демографических показателей, прогрессирующие тенденции старения экономически активного населения на селе и, как следствие, деградация производственного потенциала и связанности сельских территорий;
- недоступность (дороговизна) современных прорывных решений и технологий, в том числе цифровых;
- недостаток высококвалифицированных кадров в сельском хозяйстве и пищевой промышленности.

Таким образом, перед аграриями встает вопрос по поиску путей нейтрализации существующих вызовов и угроз. Особенно в части трансформации цепочки создания добавленной стоимости. При этом преобразование должно осуществляться постепенно, начиная с простых и доступных решений, особенно с учетом того, что для многих аграриев характерна слабая технологическая инфраструктура, высокая стоимость технологий, низкий уровень электронной грамотности и цифровых навыков.

В настоящее время цифровые технологии все активнее проникают в сельское хозяйство. Цифровизация является государственным приоритетом развития экономики, что закреплено в Федеральной программе «Цифровая экономика Российской Федерации», ведомственном проекте «Цифровое сельское хозяйство». Важнейшим фактором повышения эффективности функционирования экономики является ее цифровая трансформация [3]. Все отрасли народного хозяйства РФ, включая сельское хозяйство, активно входят в экосистему цифровой экономики. В аграрном секторе России все шире используются инновации цифрового характера, основанные на применении компьютерной техники, баз данных, проводных и беспроводных сетей, программного обеспечения с широким разнообразием алгоритмов обработки данных для принятия управленческих решений.

Экосистема цифровой экономики включает рынки и отрасли традиционной экономики, цифровые технологии и решения, нор-

мативно-правовое регулирование, инфраструктуру цифровой экономики, кадры для интенсивного развития цифровой экономики. Развитие цифровой экономики базируется на внедрении цифровых платформ как в агробизнесе, так и на государственном уровне [17].

Платформизация АПК позволит получать и обрабатывать информацию для принятия качественного управленческого решения на всех уровнях управления АПК. Процессы разработки и внедрения цифровых платформ в АПК сопровождаются различными проблемами, в том числе проблемами защиты информации и правовой защиты пользователей цифровых платформ. Основным направлениям решения данных сложностей является разработка и внедрения смарт-контракта на основе технологий распределенных реестров.

Цель данного учебно-методического пособия – дать студентам представление о цифровых платформах в АПК и работе с технологией систем распределенного реестра. Основной задачей учебно-методического пособия является описание методики создания смарт-контракта, который является важнейшим компонентом цифровой платформы.

ГЛАВА 1. ЦИФРОВЫЕ ПЛАТФОРМЫ В АПК

1.1. Цифровая платформа Министерства сельского хозяйства Российской Федерации (ЦП МСХ РФ)

ЦП МСХ РФ предназначена для обеспечения возможности получения гражданами и бизнесом комплексных государственных услуг в сфере сельского хозяйства, сгруппированных по основным жизненным ситуациям в сфере господдержки. Основной целью является цифровая трансформация сервиса предоставления мер государственной поддержки агропромышленного комплекса РФ субъектам АПК (сельхозтоваропроизводителям) и другим получателям мер государственной поддержки АПК.

ЦП МСХ РФ обеспечивает меры господдержки АПК:

для субъектов АПК, в том числе сельскохозяйственных товаропроизводителей, и других получателей:

- ускорение процессов получения и повышение результативности мер государственной поддержки,
- сокращение затрат сельскохозяйственных товаропроизводителей на предоставление отчетности;

для Минсельхоза РФ, региональных органов управления (РОУ) и муниципальных органов управления (МОУ) АПК, а также ведомственных организаций, задействованных в АПК:

- повышение объемов и качества производства продукции сельскохозяйственной отрасли РФ за счет оптимизации и облегчения доступа к предоставляемым мерам господдержки;

• повышение эффективности процессов формирования сведений о текущем состоянии АПК и прогнозе его развития;

• повышение достоверности сведений, их точности за счет автоматизации процессов проведения форматно-логического контроля данных, а также за счет минимизации ошибок человеческого фактора;

для всех вышеперечисленных участников:

• качественное улучшение информационного взаимодействия субъектов АПК и органов государственной власти за счет перевода взаимодействия в электронный формат.

В состав ЦП МСХ РФ входят следующие основные и обеспечивающие сервисы, показанные на рисунках 1.1–1.4.

К основным сервисам относятся:

1. Личный кабинет получателя мер господдержки агропромышленного комплекса в составе модулей (рисунок 1.1):

• регистрации и подтверждения данных;

• ведения сведений по основной хозяйственной деятельности;

• выбора/подбора мер господдержки;

• формирования заявок на заключение договора получения мер господдержки;

• заключения соглашений на получение мер господдержки;

• формирования дополнительных соглашений к ранее полученным мерам господдержки;

• информирования получателя МГП;

• аккредитации субъекта АПК на получение статуса СХТП;

• обучения СХТП в целях аккредитации;

• формирования персональных предложений по мерам поддержки с учетом требований региона;

• регистрации и учета коммерческих соглашений по сбыту и закупке сырья и готовой продукции;

• формирования потребности СХТП в основной деятельности.

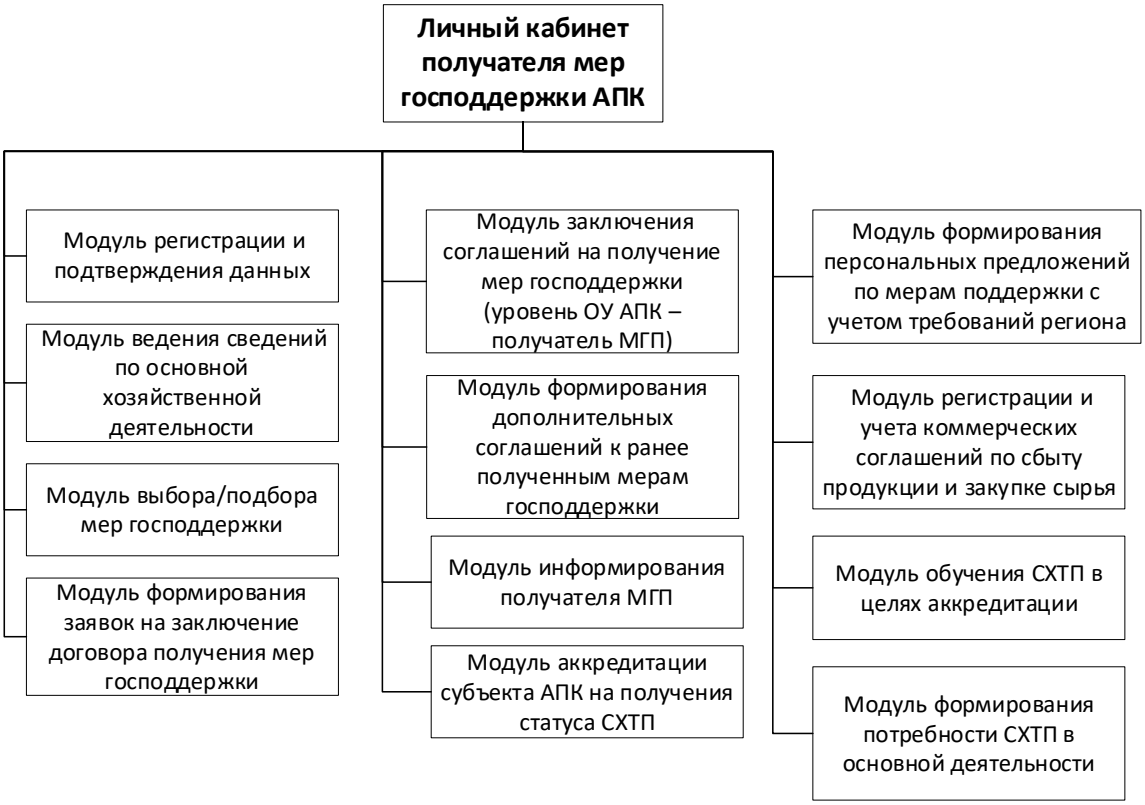


Рисунок 1.1 – Модули личного кабинета получателя мер господдержки

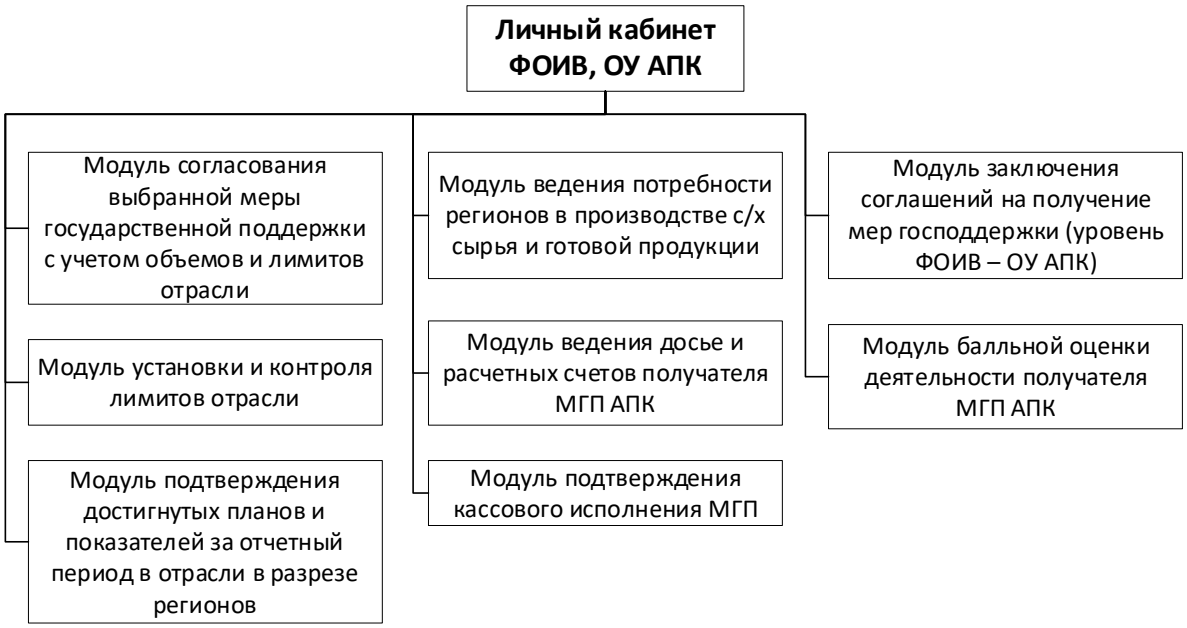


Рисунок 1.2 – Модули личного кабинета ФОИВ, ОУ АПК

2. Личный кабинет федерального, региональных и муниципальных органов исполнительной власти агропромышленного комплекса в составе модулей (рисунок 1.2):

- заключения соглашений на получение мер господдержки;
- ведения потребности регионов в производстве сельскохозяйственного сырья и готовой продукции;
- установки и контроля лимитов отрасли;
- согласования выбранной меры господдержки с учетом объемов и лимитов отрасли;
- балльной оценки деятельности получателя МГП АПК;
- ведения досье и расчетных счетов получателя МГП АПК.

Рисунок 1.3 – Обеспечивающие сервисы (модули) ИС ЦС АПК

К обеспечивающим сервисам относятся (рисунок 1.3):

- информационный портал;
- модуль интеграции с внешними источниками данных;
- группа функций обеспечения юридической значимости документов, включая модуль ведения и подтверждения юридически значимой информации;
- модуль ведения и подтверждения юридически значимой информации;
- модуль подтверждения кассового исполнения МГП;

- модуль сбора отчетности;
- группа функций обеспечения информационной безопасности;
- группа функций по ведению нормативно-справочной информации;
- конструктор мер государственной поддержки;
- группа функций аналитики и прогнозирования;
- группа функций мониторинга и диспетчеризации;
- компонент поддержки пользователей и управления инцидентами;
- модуль ведения реестра аккредитованных СХТП.

Система взаимодействует с внешними системами федерального и регионального уровня, с информационными системами органов государственной власти: ФНС, Росреестр, Роспотребнадзор, ФТС, МЧС, Россельхознадзор, Минфин, Казначейство, Минприроды, и информационными системами Минсельхоза России ФГИС ЦИАС СХ, АИС НСИ, ИС ПК ГП, СМ ПБ, ФГИС УСМТ, ЕФИС ЗСН, другими ИС, содержащими данные в машиночитаемом виде, а также предоставляет веб-интерфейс внешнего пользователя для получения гражданами и организациями данных, хранящихся в Системе, и интерфейс для получения открытых данных в соответствии с «Методическими рекомендациями по публикации открытых данных государственными органами и органами местного самоуправления версии 3.0» (рисунок 1.4).

Цифровая платформа «Государственная информационная система сбора и анализа «Единое окно» Министерства сельского хозяйства Российской Федерации представляет из себя единую аналитическую систему сбора и анализа отраслевых данных по основным, ключевым показателям/индикаторам отраслевых данных, основной целью которой является обеспечение оперативности и повышение качества сбора, обработки и анализа отраслевых данных агропромышленного комплекса Российской Федерации на основе внедрения единой точки сбора и обработки отраслевых данных.

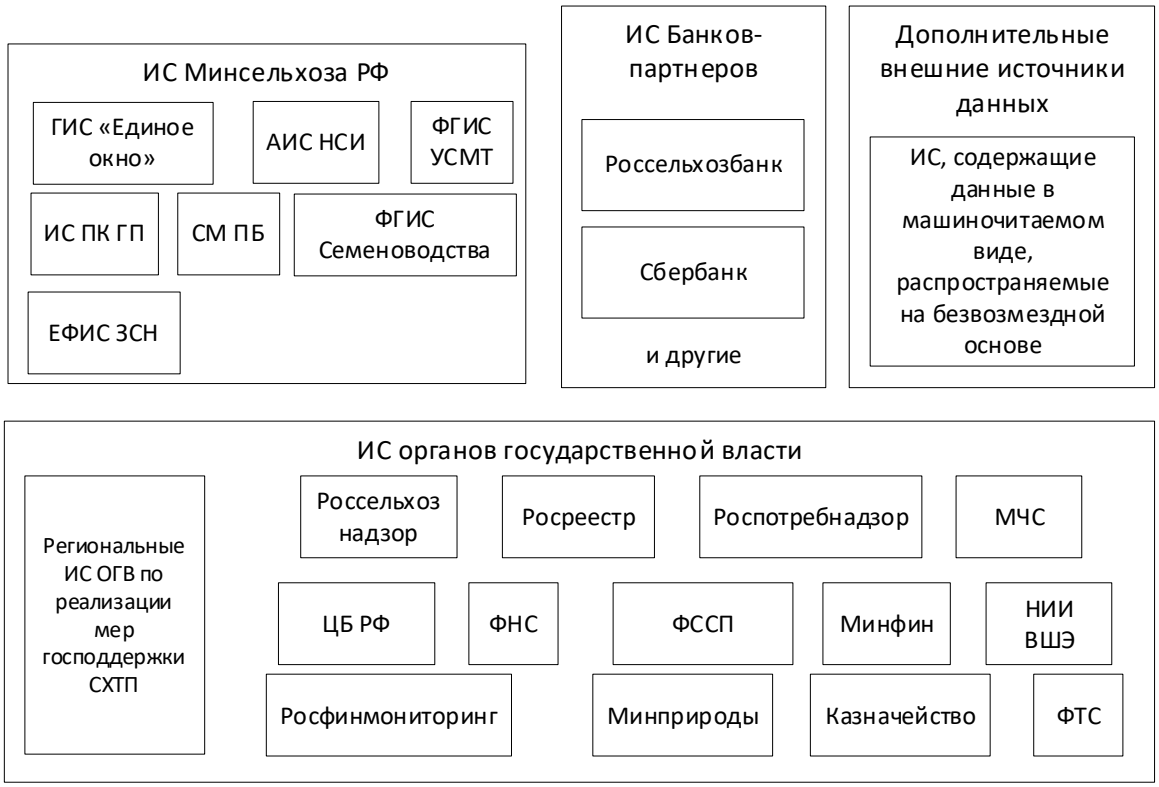


Рисунок 1.4 – Поставщики данных ИС ЦС АПК

Внедрение платформы преследует следующие цели [16]:

- повышение оперативности, объективности и качества сбора, обработки и анализа данных путем внедрения единой точки сбора и обработки статистических данных агропромышленного комплекса;
- повышение производительности труда ответственных специалистов министерства путем снижения объемов ручной обработки данных, освобождения специалистов от рутинной работы по сбору и анализу данных АПК за счет использования Системы, а также исключения практики подготовки исходных данных в электронных таблицах и текстовых редакторах;
- обеспечение непротиворечивости и единства данных, собираемых из смежных информационных систем, и повышение степени их достоверности за счет автоматической их проверки Системой;
- повышение оперативности информационного взаимодействия в электронном виде Минсельхоза России с другими государственными органами Российской Федерации в части обмена данными АПК.

Область применения: информационно-аналитическая деятельность специалистов сельского хозяйства в части мониторинга состояния агропромышленного комплекса.

Функциональные возможности: сбор, обработка и анализ отраслевых данных агропромышленного комплекса Российской Федерации, в том числе:

- обеспечение доступности аналитической информации;
- обеспечение загрузки данных из различных источников;
- обеспечение сбора данных;
- обеспечение возможности верификации значений вводимых показателей;
- обеспечение отслеживания дисциплины сбора показателей;
- обеспечение неизменности значений показателей;
- обеспечение возможности контроля исполнения регламентов верификации значений показателей и выявление фактов их нарушений;
- предоставление верифицированных значений аналитических показателей;
- анализ показателей;
- создание единого реестра отраслевых показателей и разработка на его основе единой, непротиворечивой, расширяемой метамодели данных, исключение попадания в отчетные материалы данных, не соответствующих метамодели;
- организация автоматизированного сбора данных в электронном виде из внешних информационных систем взамен ручному формированию данных;
- создание единого хранилища аналитических данных на основе единой метамодели;
- унификация и цифровизация процессов контроля предоставления отчетности и проверки полученных данных;
- унификация способов обработки, анализа, представления и методологии исследования данных.

Существующая структура функциональных подсистем и функциональных компонентов системы «Единое окно» представлена на рисунке 1.5.

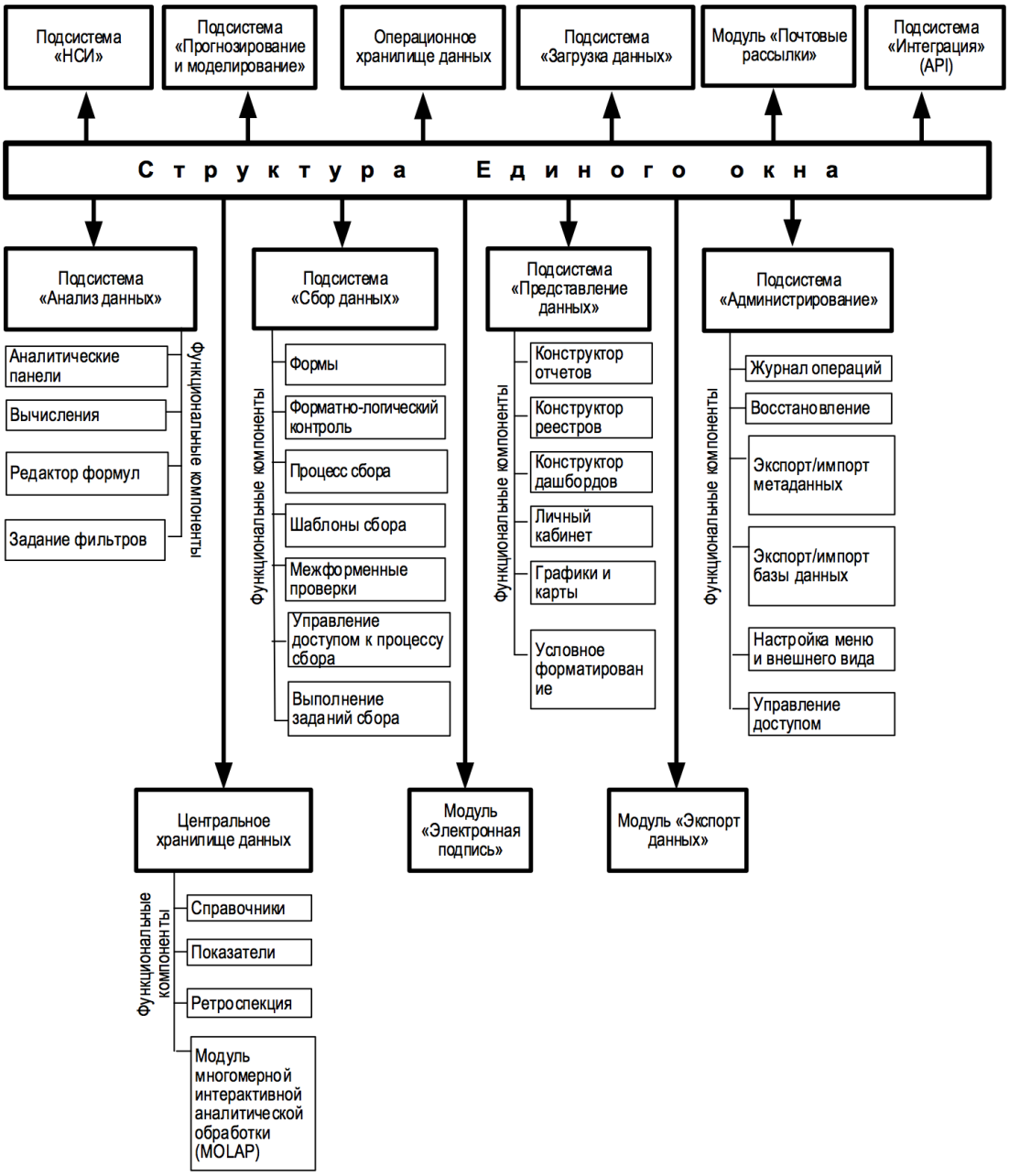


Рисунок 1.5 – Структура государственной информационной системы Минсельхоза России «Единое окно»

Состав функциональных подсистем и модулей информационной системы «Единое окно», а также реализуемых ими групп функций представлены в таблице 1.1.

Таблица 1.1 – Состав ГИС «Единое окно»

№	Подсистема /модуль	Функционал	Реализуемая компонентом группа функций
1	Анализ данных	Аналитические панели	Реализация интерфейса для выполнения анализа многомерных показателей
2	Анализ данных	Вычисления	Поддержка расчетов и работы с вычисленными значениями показателей
3	Анализ данных	Редактор формул	Обеспечение интерфейса для ввода и редактирования формул расчета вычисляемых показателей
4	Анализ данных	Задание фильтров	Обеспечение возможности настройки сложных фильтров с помощью специальных формул при конструировании реестров, форм и отчетов
5	Сбор данных	Формы	Реализация интерфейса для работы с формами сбора данных
6	Сбор данных	Форматно-логический контроль	Проверка корректности вводимых данных при заполнении форм и реестров
7	Сбор данных	Процесс сбора	Обеспечение настройки процесса сбора данных
8	Сбор данных	Шаблоны сбора	Настройка параметров процессов сбора в соответствии с установленными регламентами сбора отчетности
9	Сбор данных	Межформенные проверки	Проверка выполнения условий, связывающих данные форм и реестров определенного задания сбора
10	Сбор данных	Управление доступом к процессу сбора	Управление доступом к процессу сбора данных
11	Сбор данных	Выполнение заданий сбора	Реализация интерфейса для выполнения заданий по сбору отчетности
12	Представление данных	Конструктор отчетов	Реализация интерфейса для конструирования и представления табличных данных в виде отчетов на основе показателей и их аналитических признаков

Продолжение таблицы 1.1

13	Представление данных	Конструктор реестров	Реализация интерфейса для конструирования и представления табличных представлений данных в виде реестров данных на основе показателей и их аналитических признаков
14	Представление данных	Конструктор дашбордов	Реализация интерфейса конструирования и представления параметризованных тематических информационных панелей, обеспечивающих одновременное представление данных в разном виде (табличном, графическом, картографическом, в виде кода и т. п.)
15	Представление данных	Личный кабинет	Реализация интерфейса представления дашбордов, отчетов, реестров и т. п. в разрезе пользователей
16	Представление данных	Графики и карты	Обеспечение формирования графических диаграмм на основании аналитических выборок. Обеспечение отображения данных на карте
17	Представление данных	Условное форматирование	Обеспечение цветовой дифференциации данных и простого форматирования ячеек отчета (формы, реестра) в зависимости от заданных условий
18	Центральное хранилище данных	Центральное хранилище данных	Центральное хранилище данных
19	Центральное хранилище данных	Справочники	Ведение единых нормативных справочников и классификаторов, обеспечивающих согласованность и сопоставимость данных ГИС «Единое окно»
20	Центральное хранилище данных	Показатели	Обеспечение формирования в Центральном хранилище данных структуры взаимосвязанных показателей
21	Центральное хранилище данных	Ретроспекция	Обеспечение версионного хранения данных Центрального хранилища данных ГИС «Единое окно», просмотр и анализ исторических (ретроспективных) данных

Продолжение таблицы 1.1

22	Центральное хранилище данных	Модуль многомерной интерактивной аналитической обработки (MOLAP)	Реализация многомерного представления данных для одновременного анализа по нескольким измерениям (многомерный анализ данных)
23	Администрирование	Журнал операций	Обеспечение учета и контроля операций, выполняемых пользователями в ГИС «Единое окно»
24	Администрирование	Восстановление	Обеспечение восстановления ГИС «Единое окно» в заданное состояние путем создания точек восстановления и возврата к сохраненным точкам восстановления (при необходимости)
25	Администрирование	Экспорт/импорт метаданных	Экспорт/импорт метаданных
26	Администрирование	Экспорт/импорт базы данных	Экспорт/импорт базы данных
27	Администрирование	Настройка меню и внешнего вида	Обеспечение настройки главного меню графического интерфейса ГИС «Единое окно» и настройки внешнего вида графического интерфейса ГИС «Единое окно»
28	Администрирование	Управление доступом	Управление доступом к элементам ГИС «Единое окно»
29	Электронная подпись	–	Поддержка средств криптографической защиты информации для обеспечения юридической значимости предоставляемых (вводимых) данных
30	Экспорт данных	–	Обеспечение экспорта данных из отчетов, реестров, дашбордов во внешний файл

Продолжение таблицы 1.1

31	НСИ	–	Нормативно-справочная информация. Обеспечение непротиворечивым набором справочников, импортируемых из разных систем. Нормализация данных справочников
32	Прогнозирование и моделирование	–	Обеспечение возможности построения моделей и проведения многовариантных расчетов по заданным показателям в зависимости от комплекса управляющих параметров
33	Операционное хранилище данных	–	Вспомогательное хранилище данных. Обеспечивает в случае необходимости хранение данных, полученных из внутренних и внешних информационных систем, в их первоначальном виде
34	Загрузка данных	–	Обеспечение загрузки данных в ГИС «Единое окно» из внешних информационных систем и внешних файлов
35	Почтовые рассыл-ки	–	Обеспечение рассылки по электронной почте уведомлений пользователям
36	Интеграция	–	Обеспечение интеграции разнородных распределенных информационных подсистем за счет использования API-интерфейса

Единая федеральная информационная система земель сельхозназначения (ЕФИС ЗСН) предназначена для обеспечения деятельности МСХ и подведомственных ему учреждений и организаций актуальной и достоверной информацией о землях сельскохозяйственного назначения и землях, используемых или предоставленных для ведения сельского хозяйства в составе земель иных категорий, включая информацию о местоположении, состоянии и фактическом использовании таких земель, и состоянии сельскохозяйственной растительности.

Целями создания ЕФИС ЗСН являются:

- повышение эффективности планирования использования территорий сельских поселений и межсельских территорий, в том числе определения приоритетных направлений развития экономики соответствующих территорий;
- информационно-аналитическое обеспечение процессов подготовки и принятия управленческих решений, направленных на сбалансированное и устойчивое развитие сельскохозяйственного производства с сохранением плодородия почв, предотвращение процессов деградации земель и выбытия их из сельскохозяйственного оборота, вовлечение неиспользуемых земель в сельскохозяйственный оборот [6];
- осуществление государственного мониторинга использования и состояния земель сельскохозяйственного назначения, плодородия земель сельскохозяйственных угодий, мониторинга использования и состояния мелиоративных систем и гидротехнических сооружений, регулярная оценка состояния сельскохозяйственных культур и структуры севооборотов [14];
- консолидация сведений о землях сельскохозяйственного назначения из различных источников, в том числе на базе реализации информационного взаимодействия с другими федеральными и региональными органами исполнительной власти Российской Федерации, использования данных государственных и муниципальных информационных систем, и фондов;
- повышение скорости сбора и качества обработки информации путем автоматизации процессов сбора, обработки, анализа и предоставления получаемой информации в интересах решения задач МСХ на базе современных информационных и телекоммуни-

кационных технологий.

Система должна обеспечить решение следующих задач:

• получение, хранение, обработка, анализ объективных, актуальных и достоверных сведений о землях сельскохозяйственного назначения и землях, используемых или предоставленных для ведения сельского хозяйства в составе земель иных категорий (далее − сельскохозяйственные земли);

• учет земель сельскохозяйственного назначения, включая сельскохозяйственные и несельскохозяйственные угодья, учёт мелиорируемых земель, мелиоративных систем и гидротехнических сооружений;

• систематическое наблюдение за состоянием и использованием земель сельскохозяйственного назначения, в том числе сельскохозяйственных угодий, параметрами плодородия почв, развитием и распространением процессов их деградации, изменением состояния растительного покрова на сельскохозяйственных угодьях (пашня, сенокосы, пастбища, многолетние насаждения, залежь) [6];

• визуализация результатов государственного мониторинга земель сельскохозяйственного назначения и мониторинга земель, используемых или предназначенных для ведения сельского хозяйства в составе земель иных категорий, мониторинга плодородия почв в виде тематических карт различного содержания [16];

• отображение границ особо ценных сельскохозяйственных земель и зон их охраны [14];

• формирование статистической информации о землях сельскохозяйственного назначения [14];

• обеспечение авторизованных пользователей и заинтересованных лиц сведениями о сельскохозяйственных землях и аналитической информацией, создаваемой в ЕФИС ЗСН.

Архитектурно Система представляет собой вертикально-интегрированную структуру автоматизированных рабочих мест пользователей, в том числе в субъектах Российской Федерации, позволяющую осуществлять ведение реестра сельскохозяйственных земель в пределах территории субъекта Российской Федерации («фронт-офисы») и учет сведений о сельскохозяйственных землях на федеральном уровне («бэк-офис»). При этом Система обеспечивает возможность одновременной обработки множественных геопро-

странственных запросов, в том числе экстерриториальных, в целях предоставления сведений, содержащихся в Системе (рисунок 1.6).

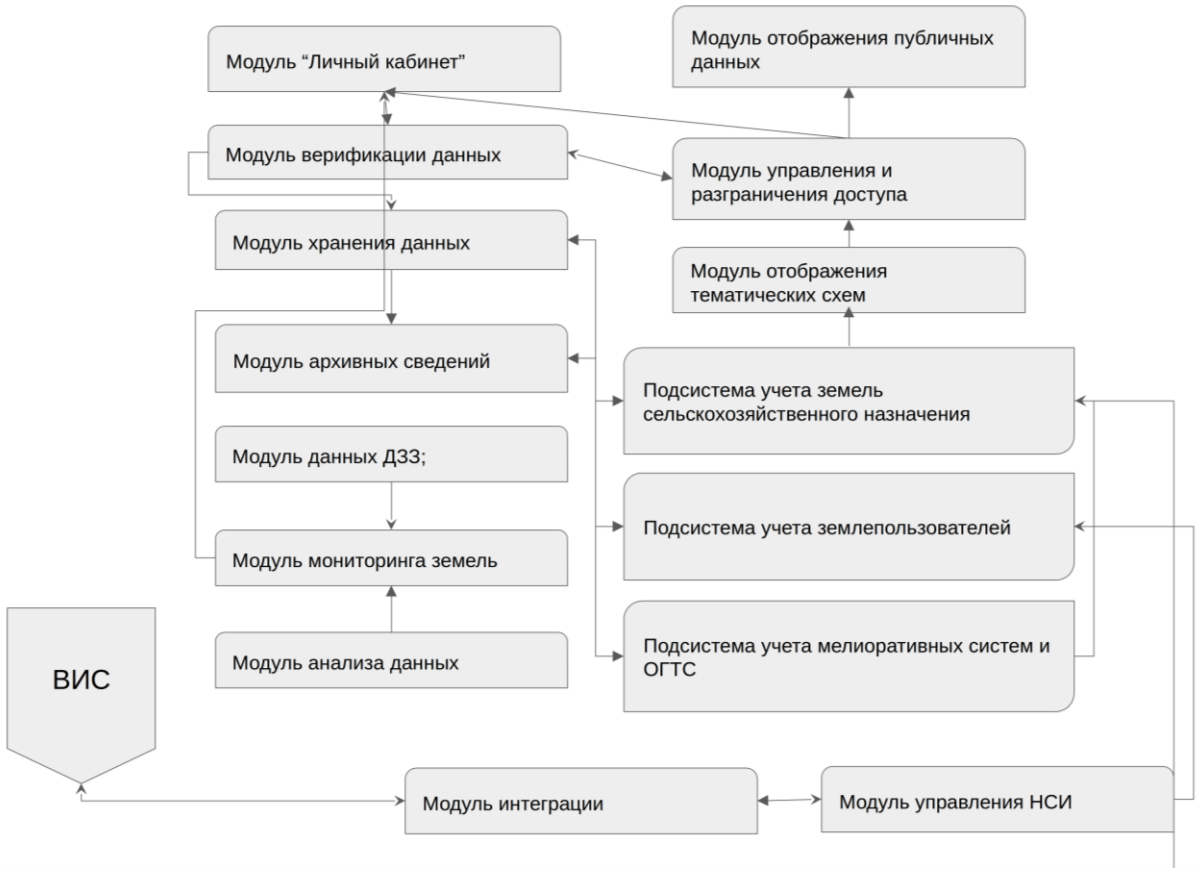


Рисунок 1.6 – Архитектура ЕФИС ЗСН

Подсистема учета сельскохозяйственных земель предназначена для учета и мониторинга ЗСН, хранения их описания (включая геометрическое). Подсистема хранит информацию о контурах.

Подсистема реализована в виде приложения «СХ Угодья». Система предоставляет доступ к объектам реестра ЗСН в рамках структуры иерархии:

- федерация в целом;
- федеральный округ;
- субъект федерации;
- муниципальное образование;
- землепользователь.

Права доступа для пользователя назначаются администратором в специальном интерфейсе управления доступом.

Пользователям доступны возможности добавления, измене-

ния, удаления объектов, изменение контуров, назначения собственников. Подсистема обеспечивает отображение данных из других подсистем в рамках объектов ЗСН.

1. Подсистема учета землепользователей.

Подсистема учета землепользователей предназначена для получения и хранения информации о землепользователях, использующих ЗСН. Подсистема реализована в виде приложения «Организации». Данные в подсистему передаются из баз данных ФНС средствами СМЭВ. Пользователю доступны для редактирования сведения об организации, цвет организации на аппликационной схеме, реестр полей предприятия.

2. Подсистема учета мелиоративных систем и отдельно расположенных гидротехнических сооружений [6].

Подсистема учета мелиоративных систем и отдельно расположенных гидротехнических сооружений предназначена для учета и мониторинга мелиоративных систем и отдельно расположенных гидротехнических сооружений. Подсистема доступна в виде приложения «Мелиоративные системы». Приложение реализует хранение и отображение (в том числе на карте) объектов мелиорации. Подсистема хранит и отображает информацию о водозаборах, водовыпусках, водотоках, водохранилищах, туннелях, трубопроводах, пунктах водоучета, плотинах, насосных станциях, каналах.

3. Подсистема загрузки данных.

Подсистема загрузки данных предназначена для осуществления загрузки данных в подсистемы ЕФИС ЗСН в формате расширенных геошаблонов. Подсистема доступна в виде приложения «Загрузка данных». Приложение реализует функциональность загрузки данных, форматно-логического контроля данных, загрузки данных в основную БД, изменений статусов наборов загружаемых данных. Подсистема позволяет загружать данные сельскохозяйственных угодий, агрохимии, мелиорации.

4. Подсистема Росреестра.

Подсистема Росреестра предназначена для структурированного отображения данных, полученных посредством СМЭВ 3 от Росреестра. Подсистема доступна в виде приложения «Росреестр». Приложение реализует функциональность просмотра данных в разрезе субъектов административно-территориального деления, функциональность просмотра правообладателей земельных участ-

ков, функциональность отображения пересечения земельных участков Росреестра и сельскохозяйственных контуров ЕФИС ЗСН.

5. Подсистема интеграции СМЭВ3.

Подсистема интеграции предназначена для обеспечения процессов межведомственного взаимодействия с использованием системы СМЭВ3. Программный продукт позволяет отправлять и принимать сообщения СМЭВ3, логировать процессы взаимодействия со СМЭВ3, осуществлять интеграцию с внутренними информационными системами пользователя. Программный продукт предоставляется в виде набора docker-образов и требует предварительной настройки развертывания этих образов, а также предварительного конфигурирования и подключения контейнера закрытого ключа.

6. Подсистема интеграции с OpenWeatherMap.

Подсистема интеграции предназначена для и связи глобальным онлайн-сервисом OpenWeatherMap. Подсистема обеспечивает получение данных о погодных условиях.

7. Подсистема защиты информации.

Система мониторинга и прогнозирования продовольственной безопасности Российской Федерации (СМ ПБ) предназначена для мониторинга и оценки показателей, характеризующих текущее и прогнозное состояние продовольственной безопасности, стратегического планирования развития агропромышленного комплекса, а также выявления рисков и угроз продовольственной безопасности Российской Федерации.

СМ ПБ предназначена для использования уполномоченными должностными лицами Минсельхоза России, заинтересованных федеральных ведомств, органов исполнительной власти субъектов Российской Федерации при решении задач мониторинга и прогнозирования состояния продовольственной безопасности Российской Федерации, формирования отчетных и прогнозных продовольственных балансов регионального и федерального уровней, совершенствования применяемых методов анализа и прогнозирования, повышения актуальности и достоверности используемых для расчетов исходных данных [11, 16].

Основными пользователями СМ ПБ являются сотрудники следующих организаций и их структурных подразделений:

1) Департаменты Министерства сельского хозяйства Российской Федерации:

- Департамент экономики и государственной поддержки АПК;
- Департамент регулирования рынков АПК;
- Департамент цифрового развития и управления государственными информационными ресурсами АПК;
- Департамент пищевой и перерабатывающей промышленности;
- Департамент животноводства и племенного дела;
- Департамент растениеводства, механизации, химизации и защиты растений;
- Департамент международного сотрудничества.

2) ФГБУ «Центр Агроаналитики».

3) Министерство обороны Российской Федерации.

4) Федеральная служба по ветеринарному и фитосанитарному надзору.

5) Органы исполнительной власти субъектов Российской Федерации, уполномоченные на взаимодействие с Минсельхозом России [15].

Для повышения эффективности решения задач, определенных Доктриной продовольственной безопасности Российской Федерации (утверждена Указом Президента Российской Федерации от 21.01.2020 № 20), СМ ПБ позволяет автоматизировать следующие процедуры (процессы) (рисунок 1.7):

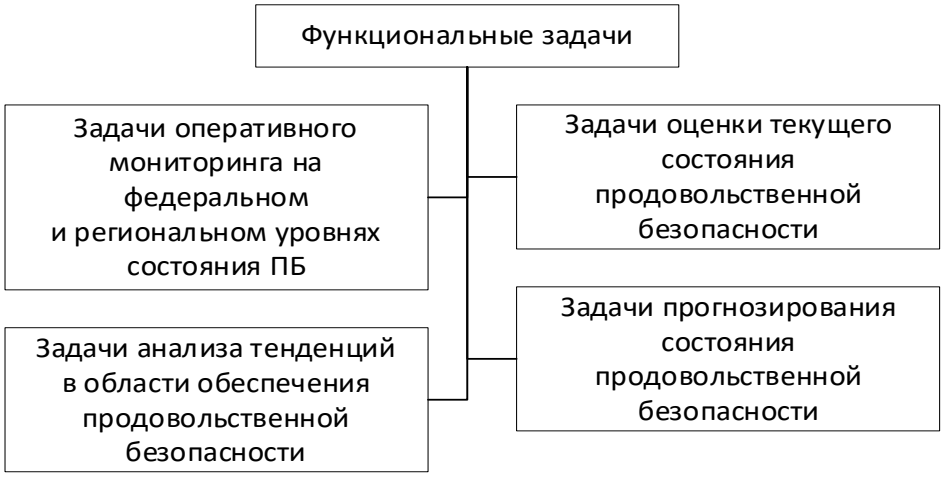


Рисунок 1.7 – Комплекс задач, возложенных на СМ ПБ

• проведение оперативного мониторинга состояния продовольственной безопасности на федеральном и региональном уровнях;

• выявление узловых проблем и анализ тенденций в области обеспечения ПБ;

• оценку текущего состояния ПБ;

• прогнозирование рисков и угроз состояния продовольственной безопасности.

В рамках СМ ПБ автоматизированы основные, обеспечивающие и управляющие процессы.

Основные процессы:

• сбор и формирование прогнозных продовольственных балансов на региональном уровне:

• сбор региональных показателей;

• мониторинг оперативных показателей;

• формирование региональных балансов продовольственных ресурсов.

Формирование и мониторинг балансов мощностей товародвижения:

• ведение базы данных объектов инфраструктуры агропромышленного комплекса, которая обслуживает потоки и запасы зерна;

• определение перспективных потребностей в мощностях инфраструктуры товародвижения в АПК по регионам.

Оценка и прогнозирование основных рисков и угроз с точки зрения их возможного воздействия на текущее и прогнозируемое состояние продовольственной безопасности Российской Федерации.

Визуализация аналитических, статистических и картографических данных, необходимых для управленческого мониторинга и принятия решений.

Интерактивная визуализация аналитических, статистических и картографических данных.

Загрузка данных по показателям деятельности АПК из внешних источников.

Обеспечивающие процессы

Администрирование СМ ПБ:

• ведение картотеки организаций;

• ведение каталога пользователей;

- предоставление доступа к информационным ресурсам.

Формирование и хранение показателей:

- ввод, расчет, уточнение и консолидация показателей;
- конструирование форм ввода данных;
- конструирование и формирование отчетов;
- обеспечение форматно-логического контроля;
- обеспечение оперативного анализа показателей.

Обеспечение поддержки сессий прогнозирования:

- обеспечение последовательности формирования показателей и балансов;
- формирование, ведение и изменение расписаний сессий прогнозирования;
- обеспечение предоставления картографического сервиса.

Обеспечение интеграции данных.

Обеспечение импорта/экспорта данных.

Управляющие процессы:

- управление контролем сбора региональных показателей;
- управление контролем исполнительской дисциплины участников сессий прогнозирования;
- управление доступом к информационным ресурсам и функциям системы.

В состав СМ ПБ входят следующие подсистемы и комплексы:

1) Функциональные комплексы:

а) комплекс оперативного мониторинга на федеральном и региональном уровнях состояния продовольственной безопасности;

б) комплекс анализа тенденций в области обеспечения продовольственной безопасности;

в) комплекс оценки текущего состояния продовольственной безопасности [16];

г) комплекс прогнозирования состояния продовольственной безопасности.

2) Прикладные подсистемы [7]:

а) подсистема работы с данными Федеральной таможенной статистики:

- функциональный блок загрузки данных Федеральной таможенной статистики – функциональный блок формирования отчетов на основании загруженных данных Федеральной таможенной ста-

тистики.

б) подсистема работы с данными ФГИС «Меркурий»:

- функциональный блок загрузки данных ФГИС «Меркурий»;
- функциональный блок формирования отчетов на основании загруженных данных ФГИС «Меркурий»;

в) подсистема прогнозирования продовольственной безопасности:

- функциональный блок ценового мониторинга (в том числе мониторинга уровня самообеспеченности);
- функциональный блок загрузки данных Росстата;
- функциональный блок формирования прогнозных продовольственных балансов;
- функциональный блок прогнозной модели цен по Российской Федерации;
- функциональный блок анализа тенденций в области обеспечения продовольственной безопасности;
- функциональный блок оценки качества и рейтингования прогнозов продовольственных балансов;

г) подсистема «Балансы мощностей товародвижения»:

- функциональный блок сбора данных по участникам зернового рынка;
- функциональный блок сбора данных по участникам молочного рынка;
- функциональный блок сбора данных по хлебофуражным балансам;
- функциональный блок картографического сервиса;
- функциональный блок отображения данных из Льготных перевозок;

д) вспомогательные функции:

- новости;
- сообщения пользователей;
- администрирование;
- НСИ;
- выгрузка данных (API).

На основе программно-технологической платформы мониторинга и прогнозирования показателей продовольственной безопас-

ности реализованы прикладные подсистемы, обеспечивающие формирование показателей продовольственной безопасности (рисунок 1.8).

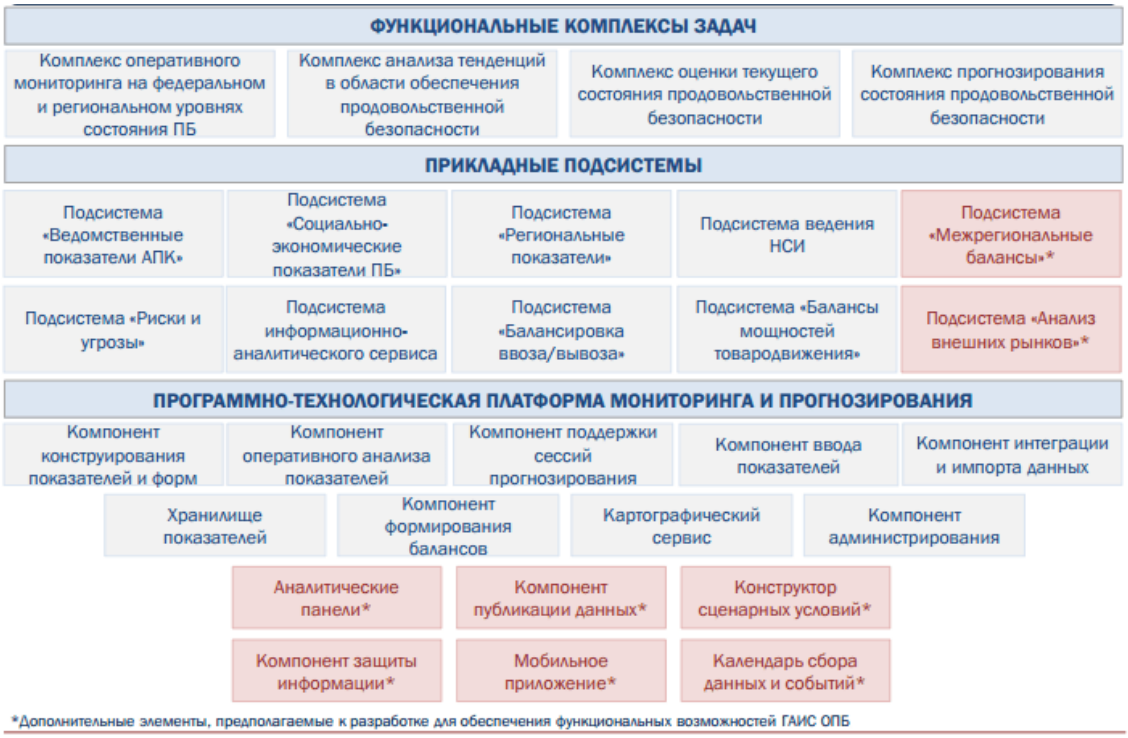


Рисунок 1.8 – Архитектура СМ ПБ

Подсистема «Ведомственные показатели АПК» предназначена для информационной, технологической и методической поддержки процессов формирования системы прогнозных продовольственных балансов Российской Федерации специалистами департаментов Министерства сельского хозяйства Российской Федерации [13].

Подсистема «Социально-экономические показатели ПБ» предназначена для информационной, технологической и методической поддержки оперативного мониторинга фактических и прогнозных показателей продовольственной безопасности, относящихся к сфере компетенции министерств и ведомств Российской Федерации, взаимодействующих с Минсельхозом России в процессе мониторинга и прогнозирования состояния продовольственной безопасности [16].

Подсистема «Риски и угрозы» предназначена для выявления и мониторинга, существующих и прогнозируемых рисков и угроз

продовольственной безопасности Российской Федерации, а также для анализа и оценки возможного воздействия на состояние продовольственной безопасности,

Подсистема «Балансировка ввоза/вывоза» предназначена для формирования прогнозных балансов продовольственных ресурсов на основе согласованных объемов взаимного ввоза-вывоза сельскохозяйственной продукции между субъектами Российской Федерации, обеспечения информационно-аналитической поддержки процессов принятия управленческих решений по вопросам участия субъектов Российской Федерации на межрегиональном рынке продовольствия и определения потенциальных возможностей для реализации своей продукции.

Подсистема «Балансы мощностей товародвижения» предназначена для обеспечения формирования прогнозных балансов мощностей инфраструктуры агропромышленного комплекса, образующих во взаимосвязи с прогнозными продовольственными балансами функциональную основу системы мониторинга и прогнозирования состояния продовольственной безопасности Российской Федерации в сферах производства и обращения продовольствия.

Подсистема информационно-аналитического сервиса предназначена для поддержки процессов анализа информации о состоянии продовольственной безопасности на уровне Министерства сельского хозяйства Российской Федерации, а также предоставления этой информации для руководителей Министерства. Пользователями Подсистемы информационно-аналитического сервиса являются руководители и специалисты департаментов Министерства [16].

Подсистема ведения нормативно-справочной информации предназначена для хранения и ведения справочников и классификаторов, используемых в подсистемах системы мониторинга и прогнозирования состояния продовольственной безопасности Российской Федерации [16].

Федеральная система прослеживаемости зерна предназначена для автоматизации процессов сбора, обработки, хранения и анализа информации о совокупности видов сельскохозяйственной и иной деятельности, связанной с производством (выращиванием зерновых культур), перевозкой, хранением, обработкой, переработкой, реализацией и утилизацией зерна и продуктов переработки зерна на внутреннем и внешнем рынках (далее − обращение зерна и про-

дуктов переработки зерна), для обеспечения прослеживаемости партий зерна и партий продуктов переработки зерна, оформления и выдачи товаросопроводительного документа на партию зерна или партию продуктов переработки зерна, внесения результатов экспертизы зерна, лабораторных исследований при ввозе на территорию Российской Федерации и вывозе с территории Российской Федерации партии зерна, обеспечения доступа к такой информации, а также для информационного обеспечения деятельности по проведению контрольных (надзорных) мероприятий при осуществлении федерального государственного контроля (надзора) в области обеспечения качества и безопасности зерна и продуктов переработки зерна, деятельности федеральных органов исполнительной власти, органов исполнительной власти субъектов Российской Федерации, органов местного самоуправления, сельскохозяйственных товаропроизводителей и других лиц, осуществляющих деятельность в области развития зернового комплекса (рисунок 1.9) [13].

Для достижения цели обеспечения учета объема партии зерна и объема партии продуктов переработки зерна при их обращении необходимо решить следующие задачи:

1) обеспечение ведения информации о товаропроизводителях;

2) обеспечение ведения реестра организаций, осуществляющих в качестве предпринимательской деятельности хранение зерна и оказывающих связанные с хранением услуги;

3) обеспечение ведения информации о партии зерна;

4) обеспечение ведения информации о партии продуктов переработки зерна;

5) обеспечение ведения информации о собственниках (владельцах) зерна, находящегося на хранении и (или) обработке, включая вид сельскохозяйственной культуры (наименование), массу (нетто в килограммах), потребительские свойства, дату принятия на хранение и (или) обработку, отгрузку, у организации, осуществляющей в качестве предпринимательской деятельности хранение зерна и оказывающей связанные с хранением услуги;

6) обеспечение ведения информации о грузоотправителях, грузополучателях, перевозчиках партии зерна и (или) партии продуктов переработки зерна;

7) обеспечение ведения информации о пунктах отправления и назначения партии зерна и (или) партии продуктов переработки зерна;

Рисунок 1.9 – Бизнес-модель формирования цепочки прослеживаемости партии зерна

8) обеспечение ведения информации о выданных СДИЗ;

9) обеспечение ведения информации о фактическом объеме (нетто в килограммах) и потребительских свойствах зерна, полученного для его первичной и (или) последующей (промышленной) переработки;

10) обеспечение ведения информации о потребительских свойствах партии зерна и (или) партии продуктов переработки зерна;

11) обеспечение ведения сведений о декларациях соответствия, фитосанитарных сертификатах, ветеринарных сертификатах на партию зерна или партию продуктов переработки зерна в случае ввоза на территорию Российской Федерации партий зерна и партий продуктов переработки зерна или их вывоза с территории Российской Федерации;

12) обеспечение ведения информации о закупке партий зерна и партий продуктов переработки зерна для государственных нужд;

13) обеспечение ведения информации о закупке партий зерна, их хранении в составе федерального интервенционного фонда сельскохозяйственной продукции и реализации в соответствии с Федеральным законом от 29.12.2006 № 264-ФЗ «О развитии сельского хозяйства»;

14) обеспечение ведения информации о документах, подтверждающих факт утилизации партии зерна или партии продуктов переработки зерна или возврат партии зерна по результатам экспертизы зерна и (или) продуктов переработки зерна, представляемых в уполномоченный Правительством Российской Федерации федеральный орган исполнительной власти, вынесший предписание о возврате партии зерна или об утилизации партии зерна или партии продуктов переработки зерна.

Для достижения цели осуществление анализа, обработки представленных сведений и информации и контроля за их достоверностью, необходимо решить следующие задачи:

- обеспечение ведения информации о результатах государственного мониторинга зерна;
- обеспечение размещения сведений и информации, содержащихся в Системе, в форме открытых данных.

В состав Федеральной системы прослеживаемости зерна включаются следующие компоненты:

•компонент ведения информации о товаропроизводителях [13];

• компонент ведения реестра юридических лиц и индивидуальных предпринимателей, осуществляющих в качестве предпринимательской деятельности хранение зерна и оказывающих связанные с хранением услуги;

• компонент ведения информации о партиях зерна;

• компонент ведения информации о партиях продуктов переработки зерна, ведения информации о партиях и собственниках зерна, находящегося на хранении и (или) обработке у юридических лиц и индивидуальных предпринимателей, осуществляющих в качестве предпринимательской деятельности хранение зерна и оказывающих связанные с его хранением услуги;

• компонент ведения информации об услугах юридических лиц и индивидуальных предпринимателей, осуществляющих в качестве предпринимательской деятельности хранение зерна;

• компонент ведения информации о грузоотправителях, грузополучателях, перевозчиках партии зерна и (или) партии продуктов переработки зерна;

• компонент ведения информации пунктов отправления и назначения партии зерна или партии продуктов переработки зерна;

• компонент ведения информации о выданных товаросопроводительных документах;

• компонент ведения информации о юридических лицах и об индивидуальных предпринимателях, осуществляющих первичную и (или) последующую (промышленную) переработку зерна;

• компонент ведения информации о государственном мониторинге зерна;

• компонент ведения информации о возврате, об утилизации, изъятии и о приостановке оборота партии зерна (продуктов переработки зерна);

• компонент ведения результатов экспертизы зерна о возврате партии зерна или об утилизации партии зерна [13];

• компонент ведения лабораторных исследований при ввозе на территорию Российской Федерации и вывозе с территории Российской Федерации партии зерна в целях оформления товаросопроводительного документа на партию зерна;

• компонент ведения информации о проведении федерально-

го государственного контроля (надзора) в области обеспечения качества и безопасности зерна и продуктов переработки зерна;

- компонент ведения информации о потребительских свойствах партии зерна и партии продуктов переработки зерна;
- компонент ведения информации о декларациях соответствия, фитосанитарных сертификатах, ветеринарных сертификатах на партию зерна или партию продуктов переработки зерна;
- компонент ведения информации о закупке партий зерна и партий продуктов переработки зерна для государственных нужд;
- компонент ведения информации о закупке и хранении зерна в федеральном интервенционном фонде сельскохозяйственной продукции, обеспечения публикации открытых данных;
- компонент интернет-портала Федеральной системы прослеживаемости зерна;
- компонент справочников и классификаторов, используемых в сфере обращения зерна и продуктов переработки зерна;
- компонент ведения сбора информации, обеспечивающий систематизацию, обработку и хранение информации, поступающей от поставщиков информации;
- компонент ведения анализа информации, обеспечивающий сопоставление и анализ информации, содержащейся в Федеральной системе прослеживаемости зерна, а также визуальные средства государственного мониторинга зерна, оценки и контроля данных о зерне и продуктах переработки зерна [13];
- компонент информационных подсистем, обеспечивающих взаимодействие с иными информационными системами, в том числе посредством единой системы межведомственного электронного взаимодействия;
- компонент информационной подсистемы обеспечения информационной безопасности [13].

1.2. Цифровые платформы в растениеводстве

Как показывает передовой практический опыт развитых стран, современное растениеводство как вид деятельности сельхозтоваропроизводителя представляет собой точное земледелие или «предписательное земледелие», основанное на использовании ин-

тегрированных систем земледелия, систем управления сельскохозяйственными предприятиями и зависит от сбора, использования, координации и анализа данных из множества источников с целью оптимизации производительности, рентабельности и устойчивости сельскохозяйственных предприятий. При этом у аграриев появляется больше эффективных инструментов на основе Big Data для принятия решений на различных уровнях (B2B, B2C) благодаря тем инструментам, которые дает ему экосистема цифрового сельского хозяйства.

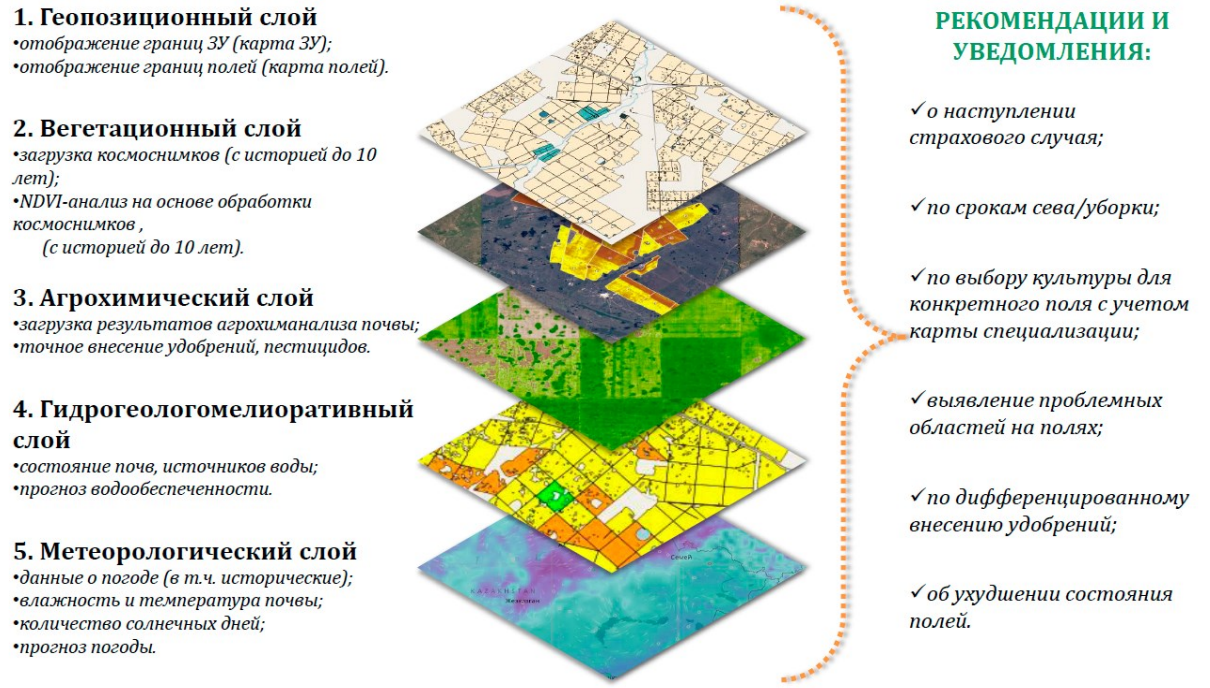


Рисунок 1.10 – Фундамент построения экосистемы цифрового сельского хозяйства в отрасли растениеводства

На российском аграрном рынке действует множество компаний – производителей и/или поставщиков цифровых решений в растениеводстве и животноводстве. Некоторые из них приведены в таблице 1.2.

Как правило, приведенные выше процессы реализуются в цифровом плане с помощью технологий интернета вещей, робототехники и сенсорики. Поэтому они, как правило, существуют как часть отраслевой цифровой системы предприятия. Такие системы называются MES-системами (Manufacturing Execution System). MES-система представляет собой облачный набор инструментов для управления производственными бизнес-процессами и контроля за

каждым производственным этапом в их взаимосвязи. О MES-системах речь пойдет ниже.

Таблица 1.2 – Цифровые технологии основных процессов в растениеводстве (точное земледелие) и животноводстве

Цифровая технология	Производитель
Растениеводство	
Параллельное вождение	АвтоГраф, Amazon, Claas Raven АгроШтурман, АгроНавигация, Trimble
Дифференцированный посев	Cognitive Agro Pilot, АвтоГраф, АгроШтурман, Cropio, Amazon, Field-IQ (Trimble), John Deere
Дифференцированное орошение	ООО Адаптивные инновационно-интеллектуальные технологии
Дифференцированное опрыскивание сорняков	Trimble, AMATRON (Amazjn), Cropio
Дифференцированное внесение удобрений	Agrofly, WeedSeeker (Trimble)
Дифференцированная обработка почвы по почвенным картам	АНТ, Геоскан, АгроДронГрупп, ГлоНАШ, ГЕОМИР
Измерение содержания хлорофилла в сельскохозяйственных культурах перед уборкой урожая	АНТ, ГЕОМИР, ЦентрПрограммСистем, Панорама
Параллельное вождение	АвтоГраф, АгроШтурман, АгроНавигация, Trimble, Amazon, Claas, Raven
Животноводство	
Управление стадом	Alta, Delaval, GEA и др.
Кормление животных	КОРАЛЛ, Delaval, GEA, SAC
Доение	Delaval, GEA, SAC, Lely
Первичная обработка молока	GEA, SAC, Lely

Цифровизация АПК, во многом благодаря развитию систем беспроводной связи, в том числе LPWAN и LORAWAN сегодня охватывает все уровни управления предприятием.

Отраслевые платформы в растениеводстве производят и продает на российском аграрном рынке множество предприятий, как отечественных, так и зарубежных. По оценкам, проводимым ООО «Агросигнал», применение интегрированных платформ приводит к

следующим эффектам:

- рост рентабельности до 20 %;
- сокращение расходов ГСМ за счет уменьшения потерь сколько (до 50 % экономии);
- рост качества выполнения работ за счет соблюдения сроков и технологических нормативов;
- сокращение непрофильного персонала на местах (бухгалтера, учетчики, диспетчеры);
- наведение порядка, структурирование информационных потоков, прозрачность производственных процессов, удобство руководителя;
- более гибкая и быстрая реакция на происходящее [12].

Сельскохозяйственные предприятия сегодня инвестируют в основном в отечественные цифровые платформы Агросигнал ExactFarming, ИнтТерра и другие. Стоимость этих цифровых платформ колеблется около 50 руб./га в год. Однако их функционал очень широк. Приведенные системы имеют приблизительно одинаковый функционал. Рассмотрим его более подробно на примере системы ANT (АО «Смарт Технологии Инвест»).

Так, например, цифровая платформа для управления сельскохозяйственным бизнесом ANT предполагает следующие возможности (рисунок 1.11).

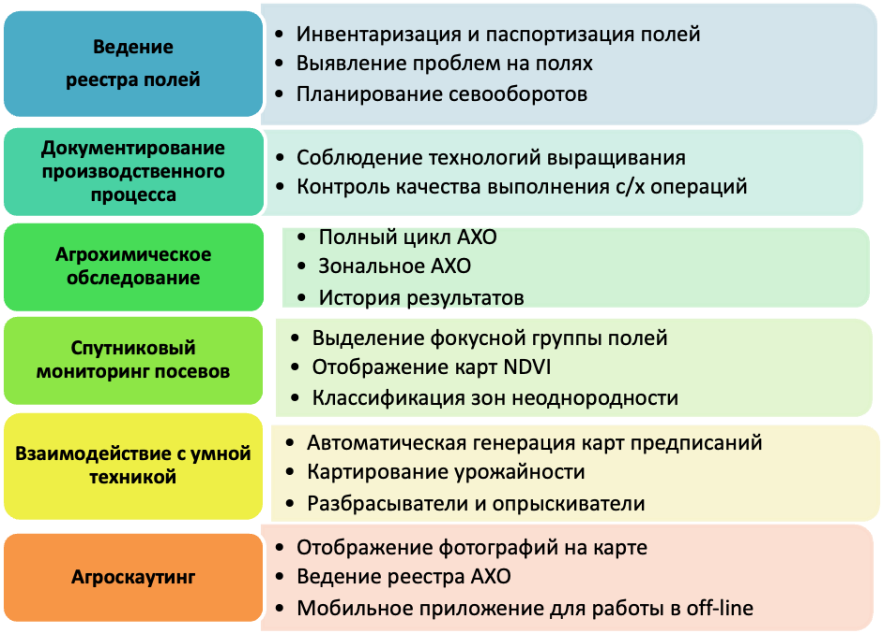


Рисунок 1.11 – Функционал интегрированной ИТ-платформы для растениеводства ANT

Система позволяет:

- осуществить гибкую настройку решений по каждому из блоков под специфику потребностей и технической оснащенности пользователя;
- быстро переработать интерфейс под конкретного пользователя (СХТП, производитель удобрений и пр.);
- комбинировать и формировать состав функциональных модулей в соответствии с потребностями пользователя.

Формирование реестра полей позволяет получить карты полей в электронном виде для организации сбора исторических данных; управления и мониторинга текущего состояния; планирования будущих сезонов.

Комплексная интегрированная информационная система управления сельхозпредприятием в растениеводстве обеспечивает учет, контроль и помощь в принятии решений по управлению. Получение оперативных и точных данные о состоянии производства дает возможность вовремя оказать воздействие, управлять ходом работ и сокращать потери в момент их возникновения, получать уведомление о простоях, нарушениях технологии выполнения работ и прочих проблемах, информирует о производительности уборочных агрегатов, объеме собранного урожая и остатках, неубранных площадях.

Основой системы ANT, так же, как и других систем, является электронная карта полей, которая позволяет актуализировать контуры полей, благодаря использованию системы ДЗЗ и проводить более адекватное планирование урожайности и объемов механизированных работ.

С помощью приложение «Скаутинг» система позволяет проводить агроскаутинг (визуальное наблюдение за состоянием участков полей) с фотофиксированием с помощью мобильного приложения участков с их географической привязкой, что позволяет удаленно управлять процессом выращивания сельскохозяйственных культур (рисунок 1.12).

Приложение «Снимки» позволяет получать карты неоднородности полей по индексу NDVI (рисунок 1.13). Также здесь загружаются высокоточные спутниковые снимки, которые могут использоваться в агроскаутинге, создании электронной карты по-

лей, для проведения зонального агрохимического обследования и карты дифференцированного внесения удобрений.

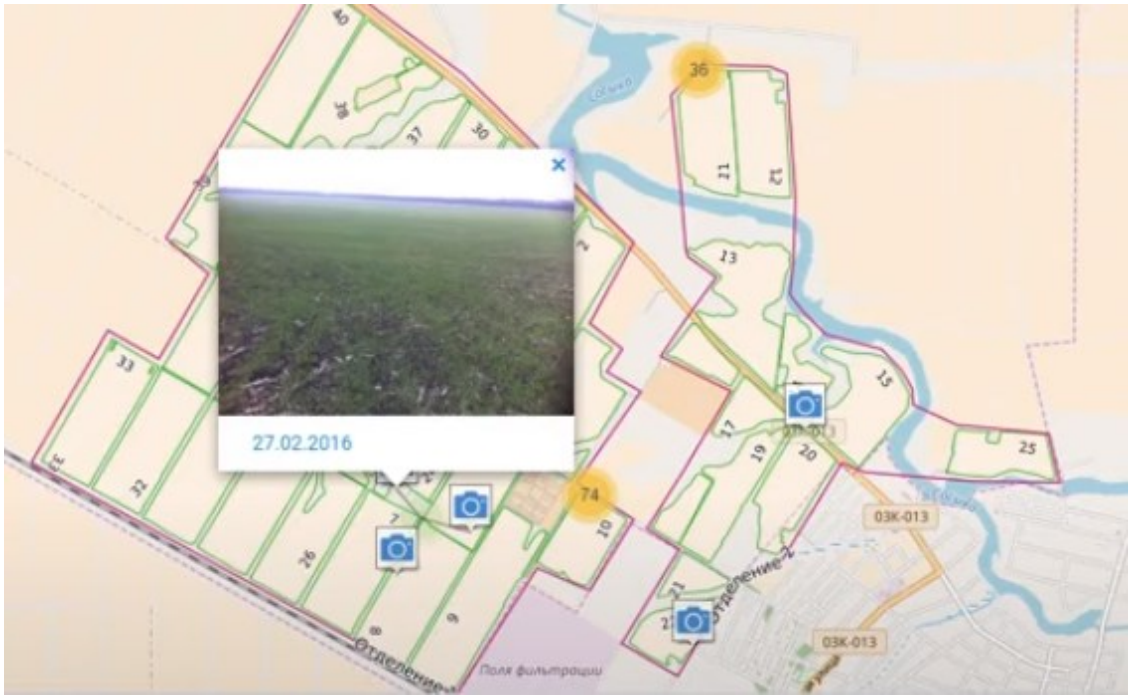


Рисунок 1.12 – Мобильное приложение ANT для агроскаутинга с географическим позиционированием фотоснимков

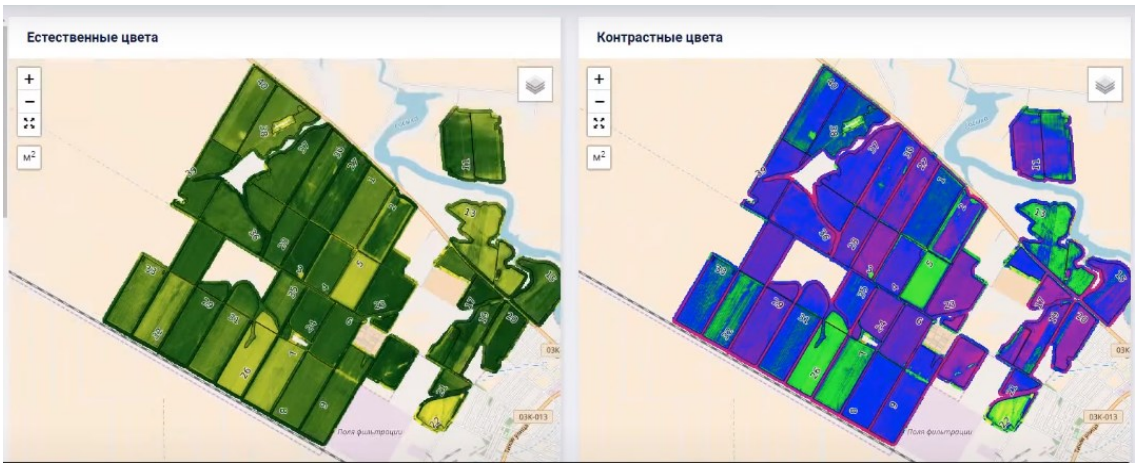


Рисунок 1.13 – Приложение системы ANT «Снимки»

Приложение «Метео» служит для получения информации с «умных» метеостанций, наличие которых позволяет проводить корректировку плана мероприятий. Данные доступны в графической и табличной форме.

Приложение «Сезоны» позволяет оптимальным образом проводить распределение культур в севообороте по полям (рисунок

1.14) и планировать затраты в зависимости от принятых технологий возделывания сельскохозяйственных культур в севообороте.

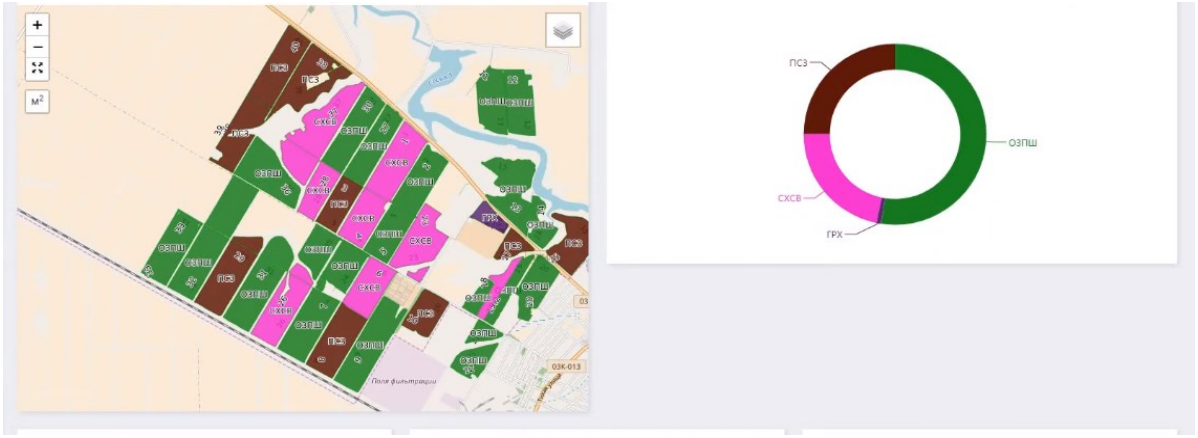


Рисунок 1.14 – Приложение системы ANT «Сезоны»

Приложение «Агроблокнот» позволяет вести учет выполненных работ, контролировать их очередность, сравнивать план и факт.

Приложение «Паспорт поля» позволяет получить полную информацию о данном поле с географической привязкой к местности (рисунок 1.15).

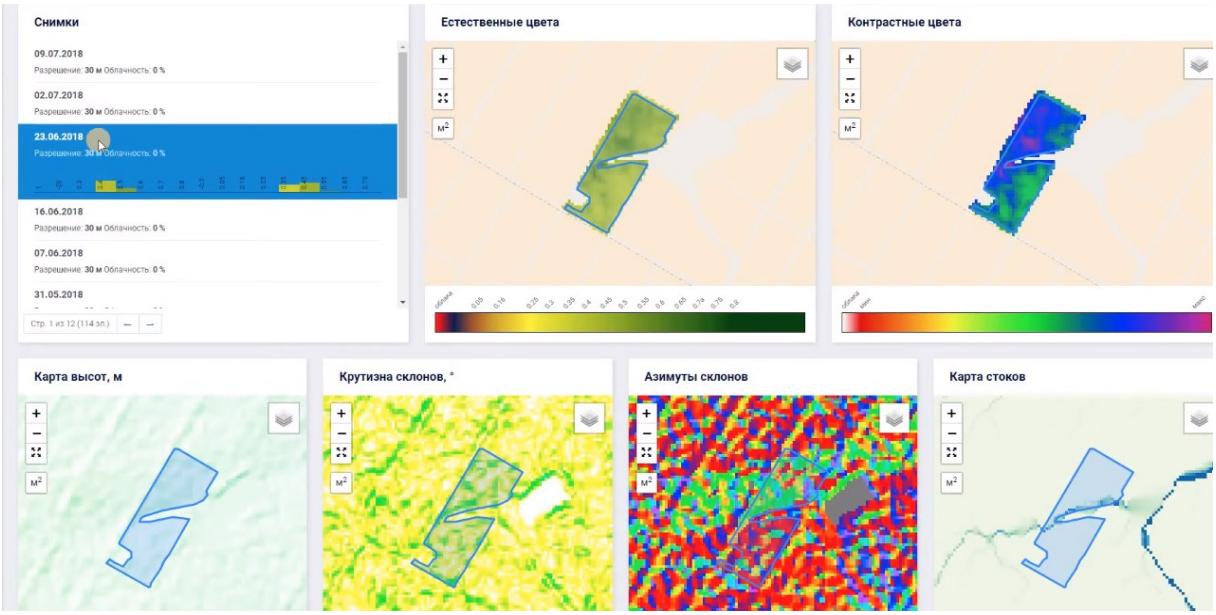


Рисунок 1.15 – Приложение системы ANT «Паспорт поля»

Там содержится информация о результатах проведения агрохимического обследования поля, ретроспектива высеваемых на нем культур, метеосводка, различные NDVI-снимки.

Приложение «Рейтинг культур» показывает наиболее урожайные для данного поля культуры. Данные наблюдений за определенный период времени за каждым полем накладываются друг на друга для определения ее средней урожайности и наличия биомассы (по индексу NDVI) на все календарные сроки для мониторинга роста и развития культуры. Данные обновляются ежедневно, что позволяет агроному постоянно контролировать состояние посевов по количеству биомассы. Это дает возможность оперативно выявлять проблемы развития растений и принимать адекватные меры для их устранения.

Приложение «Карты дифвнесений» позволяют определить на каждом из полей места для точного внесения удобрений. Приложение «БПЛА» позволяет загружать в систему снимки с беспилотных летательных аппаратов. С их помощью можно вовремя фиксировать распространение заболеваний растений, целенаправленно вносить удобрения и средства защиты растений. Приложение «Путевые листы» автоматически генерирует эти документы, исходя из оперативного плана работ. Затем в путевые листы автоматически заносится фактическая выработка конкретного работника.

1.3. Цифровые сервисы в животноводстве

В технологиях точного животноводства используются технологические процессы принципы автоматизации животноводства, позволяющие фермерам следить за здоровьем и благополучием больших популяций животных реальном масштабе времени, а также своевременно обнаруживать проблемы возникающие в процессе их жизнедеятельности и превентивно устранять их.

Технологии точного животноводства позволяют проводить неинвазивный мониторинг отдельных параметров тела животного (температура, окраска поверхности тела, слизистых оболочек и др.), выявления отдельных видов симптомов заболеваний (респираторных, инфекционных), идентификации конкретных особей (при помощи машинного зрения), идентификация отдельных физиологических состояний животного (охота, беременность и др.) и конечно поведения животных, обнаружение стрессовых ситуаций (агрессивное поведение, наоборот очень вялое поведение, не характерные поведенческие реакции и др.).

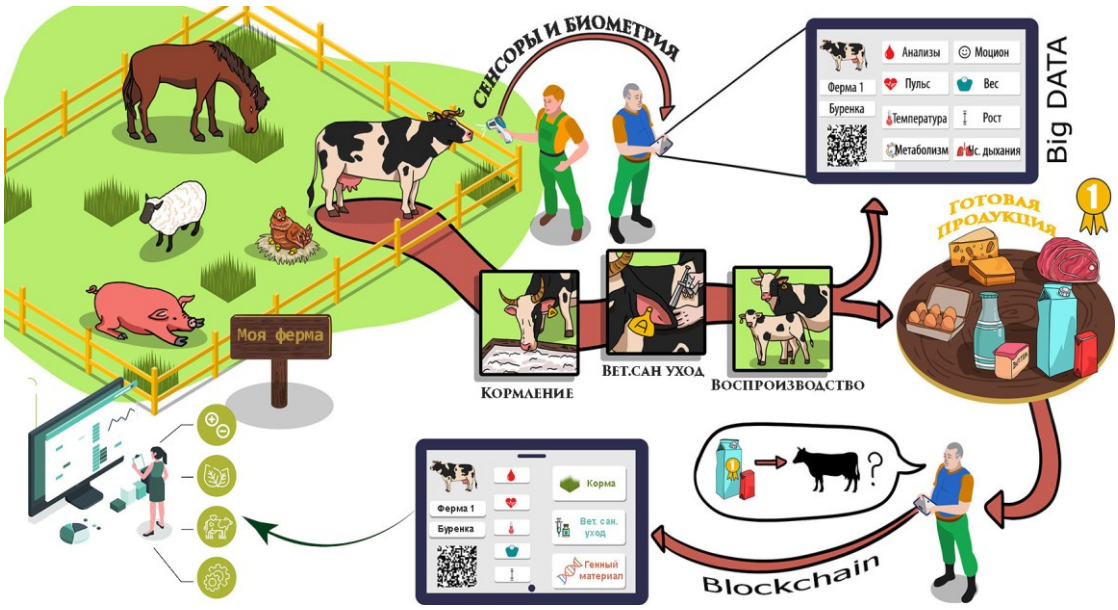


Рисунок 1.16 – Экосистема точного животноводства на базе цифровых технологий

В рамках национального проекта «Цифровое сельское хозяйство» предполагается создание и внедрение модуля «Агрорешения», предусматривающего разработку системы обеспечения операционной деятельности и внедрения комплексных цифровых решений в животноводстве: интернет вещей (IoT), «Умная ферма», «Умное стадо», в том числе с использованием технологий радиочастотной идентификации, датчиков жизнедеятельности и возможностью сбора данных из беспроводных LORA-сетей, реализации сквозных бизнес-процессов (включая модули «Мониторинг рабочего и продуктивного скота и продукции животноводства»)

Базовыми цифровыми технологиями, применяемые в животноводстве, являются большие данные (Big Data), искусственный интеллект и нейротехнологии, технологии беспроводной связи, интернет вещей (IoT), технологии робототехники и сенсорика.

Наиболее часто используемая цифровая технология в животноводстве – это интернет вещей (IoT). Примеры использования IoT-технологии в животноводстве:

- переход с пластиковых ушных бирок коров на электронные RFID-метки. RFID-метки и сканер позволяют сделать гораздо проще – найти в стаде из сотен коров одну нужную, чтобы провести процедуру (например, прививку). Цифровая идентификация позво-

ляет быстро получить полное досье коровы: от даты рождения и родителей до всех проведенных и запланированных ветеринарных мероприятий, надоях и так далее;

- существуют датчики, способные выявить «охоту», спрогнозировать пол будущего потомства, объем потребленного корма и прочие полезные для заводчиков вещи;
- терминалы для считывания данных с бирок – это часть общей системы, ядром которой выступает интегрирующая все данные учетно-аналитическая система, как правило, в РФ – на базе «1С».

В животноводстве с учетом особенностей отрасли в основном применяются следующие цифровые решения:

- персонализация животных (оценка активности, руминации, продуктивности);
- управление селекцией и ветеринарными мероприятиями (контроль охоты, состояние здоровья, планирование технологического цикла);
- кормление (контроль расхода кормов, формирование индивидуальных рационов в зависимости от особенностей и половозрастной категории животного);
- доение и первичная обработка молока (оценка качества и количества поступающего молока);
- регулирование микроклимата (поддержание оптимального температурного и газового состава воздуха в помещении, где содержатся животные).

Наиболее распространенные цифровые решения в животноводстве представлены на рисунке 1.17.

Внедрение как отдельных элементов, так и комплексных цифровых платформ имеет безусловно разную эффективность, но в целом позволит аграриям более рационально вести мониторинг использования факторов производства (состояние почвенного плодородия, процессов внесения удобрений и средств защиты растений, состояние и продуктивность животных), в реальном масштабе времени получать финансово-экономические показатели деятельности предприятия (затраты, прибыль) в расчете на единицу получаемой продукции.

Рисунок 1.17 – Наиболее распространенные цифровые решения в животноводстве

Сегодня на рынке присутствуют персональные цифровые решения для животноводства следующих производителей (таблица 1.3) [15].

Таблица 1.3 – Персональные цифровые решения для животноводства

Технология	Производитель
Управление стадом	DairyComp 305, UNITRAC и др.
Кормление животных	Коралл, Delaval, Cosmix и др.
Доение	DairyProQ, Delaval и др.
Первичная обработка молока	Nautilus (Lely), Delaval DX, BluGenium (GEA) и др.

Общим трендом является переход от отдельных технологических и отраслевых к корпоративным интегрированным информационным системам управления сельскохозяйственным производством (ERP-системам).

ERP-системы включают следующие взаимосвязанные модули: управление производством, финансовый учет, управление персоналом, управление поставками, бизнес-аналитика, управление продажами, управление конструкторской работы, планирование производства, управления закупками, управление запасами. В российском АПК в основном действует две корпоративные системы –

SAP и 1С:ERP АПК. Лидером является 1С:ERP АПК. Она внедрена на таких предприятиях, как ЭкоНива-АПК Холдинг, Агрохолдинг «Красный Восток Агро», Группа компаний «Заречное», холдинг ООО «Русская земля» (Группа компаний РЗ Агро)», АО «Агрокомплекс «Мансурово», ОАО «Русский аграрный дивизион», Группа компаний «Зеленая Долина», АО «Алексеевский Бекон», ООО «Башкирская мясная компания», ОАО «Белгородский бекон», АО «Птицефабрика Зеленецкая», АО «Свинокомплекс «Уральский», ООО «Свинокомплекс Хвалынский» и др. Система SAP внедрена в ГК РУСАГРО [15].

Популярной специализированной системой для животноводства является система DairyComp 305, которая включает автоматизацию всех бизнес-процессов фермы по основным направлениям:

- воспроизводство (позволяет синхронизировать все процессы: контроль охоты, гормон-программу, осеменение и УЗИ. Важно исключить пропуски или задваивание животных);
- доение (включает в себя ряд отчетов и графиков, позволяющих проанализировать процесс доения: общие параметры, соблюдение протокола доения, контроль ошибок персонала и неисправностей доильного оборудования);
- ветеринария (ветеринарные протоколы экономят время на введении информации и исключают ошибки в работе ветеринарных врачей. Позволяет проанализировать эффективность схем и препаратов);
- группировка (анализ своевременности технологических переводов из группы в группу. Контроль группировки дойных секций на соответствие стратегии кормления. Соблюдение периода сухостоя);
- вакцинация (создает автоматические списки, которые исключают пропуск животных при вакцинации. Информирует − когда, в каком возрасте и на каких сроках стельности была сделана та или иная вакцинация);
- молодняк (отслеживание роста и веса молодняка, контроль сохранности, а также аналитика причин выбытия. Помогает оптимизировать программу выращивания и получить нетель максимально быстро).

Сквозные цифровые технологии в животноводстве начали использоваться раньше, нежели в растениеводстве, так как многие подотрасли животноводства имеют полупромышленный характер, где применение беспроводной связи, всевозможных датчиков, сенсоров по сравнению с растениеводством облегчено. На этом основано применение в животноводстве информационно-аналитических систем, интернирующихся с установленными автоматизированными информационными системами управления фермой, которые охватывают процессы кормоприготовления, раздачи корма, управления доением, воспроизводством стада и т. д. На российских фермах действуют автоматизированные системы таких фирм, как DairyComp 305, Delaval, SDR и другие (рисунок 1.18).

Рисунок 1.18 – Производители автоматизированных систем управления животноводческой фермой

Представителем информационно-аналитических систем в животноводстве является система Dairy Production Analytics (DPA), которая позволяет с помощью специальных коннекторов подключаться к автоматизированным системам управления фермами и получать оттуда необходимую для последующей обработки и аналитики информацию. Данные также собирают специализированные датчики и сенсоры, которые замеряют температуру и влажность воздуха внутри фермы и снаружи, объем потребляемой животными воды, различные параметры готовности силоса и т. д.

Таким образом, в DPA аккумулируются следующие параметры работы фермы:

- эффективность доения;
- рационы и эффективность кормления;
- состояние здоровья каждой коровы;

- структура стада и процессы воспроизводства стада.

Применение системы DPA позволяет осуществлять следующее:

1. Онлайн-мониторинг производства молока.
2. Онлайн-мониторинг стада (воспроизводство, болезни, выбытие).
3. Выявление малопродуктивных коров для выбраковки.
4. Выявление факторов, влияющих на производство молока.
5. Прогнозирование производства молока и поголовья.
6. Построение системы мотивации персонала.
7. Построение различных прогнозов.

Комплексный подход позволяет увидеть целостную картину интеграции технологий и бизнеса предприятия и, по сути, создать *цифровой двойник* животноводческой фермы. Система позволяет строить прогнозы, период прогнозирования составляет 1,5–2 года и более, а точность прогнозов достигает 98 %.

Целью использования DPA является проведение аналитики бизнес-процессов, вопросов организации труда и производства. После нахождения «слабого звена» в работе фермы даются рекомендации по совершенствованию и перестройке бизнес-процессов. Основным критерием предлагаемой перестройки (реинжиниринга) бизнес-процессов является экономическая эффективность производства молока. Использование системы позволяет оценить, правильно ли работает ферма в рамках целевых показателей, можно ли оптимизировать какие-то показатели и повысить эффективность, принять решение, оставаться ли предприятию в рамках текущей бизнес-модели или что-то нужно изменить. На данном этапе работы применяется управленческий консалтинг – руководителям организации даются соответствующие рекомендации.

DPA как инструмент представляет собой набор динамических, визуализированных отчетов с большим количеством параметров и фильтров, например: Надои, Надои по секциям, Надои по дойкам, Поголовье, Болезни, Выбытие, Причины выбытия, Лактационные кривые, Коровы, Отелы, Прогноз погоды, Надой и персонал, Надои и температура, Доение и потери, Корма, Корма по секциям, Оплодотворяемость и другие.

Остановимся на некоторых из них более подробно.

Отчет по «Надои» позволяет проследить динамику и тренд суточного надоя на 1 корову (рисунок 1.19).

Рисунок 1.19 – Отчет по суточным надоям на корову

Динамика показателей отображается на специальной панели (рисунок 1.20).

Рисунок 1.20 – Панель оценки динамики надоев

Информация касается динамики надоев на дойную и фуражную корову, суммарной численности поголовья, количества отелов и абортов. Красная стрелка показывает снижение показателей, зеленая − рост. Также информация может быть представлена в графической форме (рисунок 1.21).

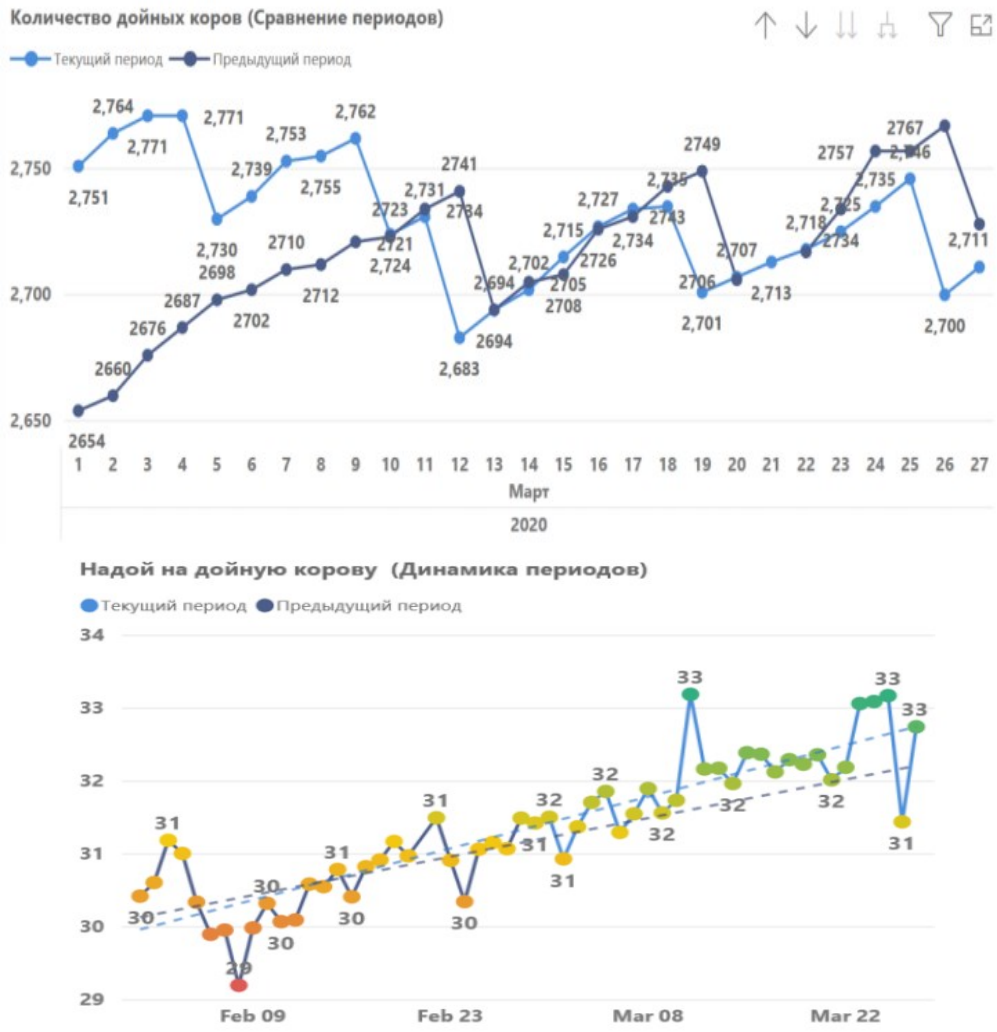


Рисунок 1.21 – Графики динамики количества коров и надоев

Здесь отражается информация по среднему надою, среднее количество дней по каждой лактации. Также есть возможность фильтрации показателей, если система установлена на нескольких фермах. Также информация может быть показана относительно каждого загона, секции. Это можно сделать в разрезе месяцев.

Аналитика кормления

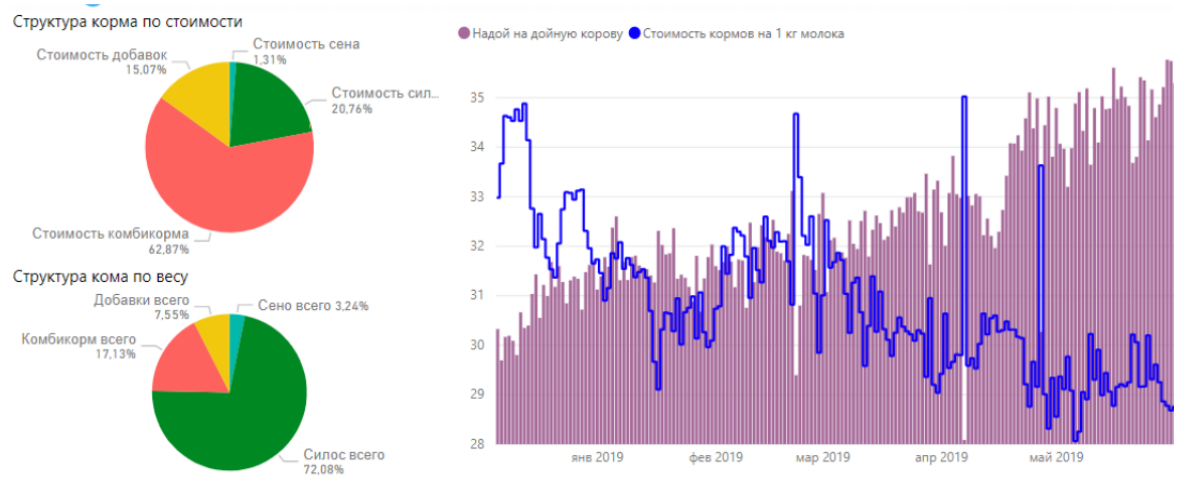
Система позволяет провести анализ качества рационов корм-

ления скота и стоимости рациона. На рисунке 1.22 представлен фрагмент отчета по стоимостной структуре рациона.

Отчет по болезням животных

Он позволяет выявить изменения в заболеваемости коров по месяцам и видам болезней, сгруппировать количество выбывших коров по группам болезней. Информация также может быть представлена в графической форме (рисунок 1.23).

Существуют также отчеты, связанные с анализом деятельности персонала. Например, отчет «Надой и персонал»: графики работы кормораздатчика и персонала синхронизируется с модулями. Это позволяет выявить факторы, которые отрицательно повлияли на надои в разрезе фамилий обслуживающего персонала. Следствием будет являться выявление причин сбоев надоев по коровам, обслуживаемым каждым из дояров (рисунок 1.24).



Год	2018	2019							
	Декабрь	Всего	Январь	Февраль	Март	Апрель	Май	Июнь	Всего
Корма всего	4 030 766	**4 030 766**	4 780 589	4 364 012	4 589 354	4 406 647	4 646 393	596 462	**23 383 456**
Сено всего	141 530	**141 530**	158 713	134 365	153 592	139 821	141 254	18 046	**745 791**
Силос всего	2 992 290	**2 992 290**	3 559 110	3 208 145	3 262 193	3 084 534	3 237 854	415 999	**16 767 836**
Комбикорм всего	720 292	**720 292**	813 019	740 702	783 868	745 702	791 190	101 326	**3 975 808**
Добавки всего	176 653	**176 653**	249 747	280 800	389 700	436 589	476 094	61 091	**1 894 021**
Стоимость кормов, руб.	27 307 581	**27 307 581**	31 755 322	29 789 553	32 418 278	31 456 560	33 372 005	4 272 501	**163 064 220**
Количество животных расчетное	107300	**107300**	125804	116168	132006	123035	132892	17541	**647446**
Количество животных DairyComp	122944	**122944**	149533	135778	150986	147472	153948	20022	**757739**
Корма на 1 гол. РАЦИОН кг/гол./сутки	37,57	**37,57**	38,00	37,57	34,77	35,82	34,96	34,00	**36,12**
Корма на 1 гол. ФАКТ кг/гол./сутки	32,79	**32,79**	31,97	32,14	30,40	29,88	30,18	29,79	**30,86**
Стоимость кормов на 1 кг молока	17,09	**17,09**	15,99	16,49	15,84	15,57	15,13	14,82	**15,75**
Стоимость кормов дойных коров на 1 кг молока	13,56	**13,56**	12,99	13,78	12,90	12,67	12,19	11,90	**12,84**
Стоимость комбикорма и добавок дойных коров на 1 кг молока	11,17	**11,17**	10,65	11,35	10,77	10,59	10,21	9,96	**10,67**
Стоимость комбикорма и добавок фуражных коров на 1 кг молока	11,85	**11,85**	11,19	12,01	11,47	11,37	10,97	10,74	**11,36**
Надой на дойную корову	30,88	**30,88**	31,62	31,76	32,44	33,13	34,71	35,32	**32,82**

Рисунок 1.22 – Отчет по динамике стоимости рациона скота в системе DPA

Рисунок 1.23 – Графический отчет по количеству выбывших животных по группам болезней (внутренние органы, болезни ЖКТ, болезни вымени и т. д.)

Дояры

Год	Блинова Ю.Н. Казакова А.И.	Казакова А.И. Михайлов А.А.	Михайлов А.А. Блинова Ю.Н.	Всего
2018	**31,34**	**31,50**	**31,49**	**31,44**
Январь	31,91	32,44	31,91	**32,12**
Февраль	30,86	30,44	30,73	**30,70**
Март	31,54	31,95	32,15	**31,88**
Апрель	33,03	33,22	33,11	**33,12**
Май	32,33	32,57	32,94	**32,61**
Июнь	33,01	33,25	33,12	**33,13**
Июль	32,79	32,45	32,45	**32,57**
Август	32,30	32,52	32,48	**32,43**
Сентябрь	30,80	31,26	31,35	**31,14**
Октябрь	28,59	28,74	28,74	**28,69**
Ноябрь	28,46	28,88	28,69	**28,67**
Декабрь	30,58	30,60	30,74	**30,64**
2019	**31,52**	**31,43**	**32,10**	**31,67**
Январь	31,52	31,43	32,10	**31,67**
Всего	**31,35**	**31,50**	**31,51**	**31,46**

Кормораздатчики

Год	Букарин	Горбунов	Всего
2018	**31,46**	**31,43**	**31,44**
Январь	32,08	32,15	**32,12**
Февраль	30,63	30,76	**30,70**
Март	31,92	31,84	**31,88**
Апрель	33,23	33,03	**33,12**
Май	32,63	32,59	**32,61**
Июнь	33,01	33,24	**33,13**
Июль	32,73	32,42	**32,57**
Август	32,42	32,44	**32,43**
Сентябрь	31,16	31,12	**31,14**
Октябрь	28,75	28,63	**28,69**
Ноябрь	28,73	28,61	**28,67**
Декабрь	30,69	30,60	**30,64**
2019	**31,65**	**31,70**	**31,67**
Январь	31,65	31,70	**31,67**
Всего	**31,47**	**31,44**	**31,46**

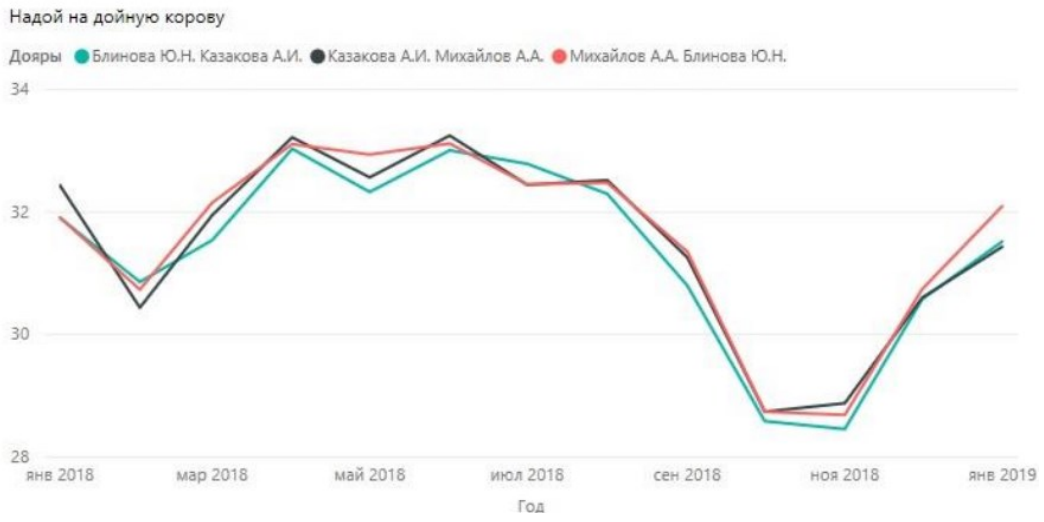


Рисунок 1.24 – Отчет по KPI

Отчет «Прогноз погоды» позволяет увидеть динамику температуры и влажности воздуха на ферме и за ее пределами, а также прогноз данных показателей (рисунок 1.25), что позволяет принимать решения по изменению содержания и кормления животных в зависимости от климатических условий.

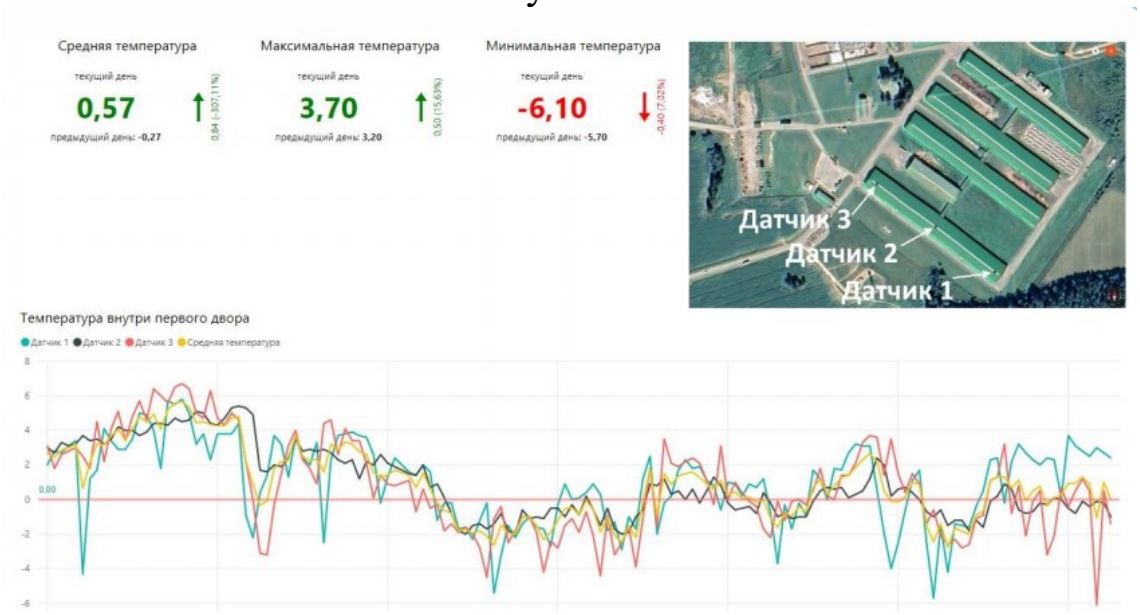


Рисунок 1.25 – Графический отчет системы DPA по динамике температурных данных

Таким образом, DPA предоставляет данные по работе фермы в различных разрезах, а также генерирует новые полезные данные в виде сводных отчетов, трендов и прогнозов. На основе этих данных топ-менеджмент фермы может принимать верные управленческие решения, которые влияют на повышение эффективности предприятия.

ГЛАВА 2. АРХИТЕКТУРА, ПОРЯДОК РАЗРАБОТКИ И ПРОЕКТИРОВАНИЯ ЦИФРОВОЙ ПЛАТФОРМЫ ДЛЯ СОВЕРШЕНИЯ ТОРГОВО-ЗАКУПОЧНЫХ ОПЕРАЦИЙ

2.1. Технология систем распределенного реестра

В последние годы российское сельское хозяйство переживает значительный рост и становится лидером российского экспорта, ведущей отраслью по обеспечению импортозамещения благодаря тому, что некоторые крупные российские аграрные предприятия активно внедряют передовые цифровые технологии в практику ведения сельского хозяйства. В настоящее время государству необходимо поощрять те крупные компании, которые начали внедрять цифровые технологии, а также стимулировать развитие сельскохозяйственной экосистемы, для того чтобы малые и средние фермерские хозяйства могли использовать цифровые технологии для преобразования моделей производства и предоставления услуг. В качестве предпосылок создания цифровой платформы можно отметить то, что существующая система совершения торгово-закупочных операций в АПК приводит к возникновению дополнительных трансакционных издержек. Сложность получения актуальной информации по сделкам и транзакциям увеличивает стоимость принятия управленческих решений. Также стоит выделить тот факт, что отсутствие стандартов и инфраструктуры для применения распределенных реестров препятствует интеграции этой технологии в бизнес-процессы предприятий АПК.

Разработка цифровой платформы реализует технологические стандарты и инфраструктуру распределенных реестров для снижения издержек и повышения прозрачности операций. Принципами построения системы является идентификация участников и контроль доступа в сеть, применение умных контрактов (смарт-контрактов) для автоматизации бизнес-процессов, использование стандартов криптографии и управление доступом к конфиденциальной информации.

Технология систем распределенного реестра представляет собой новый подход к созданию баз данных, ключевой особенностью которого является отсутствие единого центра управления. Каждый узел составляет и записывает обновления реестра независимо от других узлов [4].

В отличие от распределенных баз данных, каждый участник системы распределенного реестра хранит всю историю изменений и валидирует добавление любых изменений в систему с помощью алгоритма, который математически гарантирует невозможность подделки данных. Ни один участник взаимодействия не может изменить данные в системе таким образом, что другие участники не узнают об этом. Благодаря этому данные, которые находятся внутри системы распределенного реестра, становятся доверенными, а все изменения – прозрачными [4].

Процесс определения субтехнологий состоит из нескольких этапов, включающих в себя анализ и интерпретацию открытых источников, анализ ландшафта технологии, семантический анализ, процессы обсуждения и согласования перечня с экспертами рынка и ведущими исследователями технологии. «Субтехнология» является компонентой «сквозной» цифровой технологии, благодаря которой возможно выделение продуктов и решений и их отнесение к данной «сквозной» цифровой технологии. Каждая субтехнология является уникальной компонентой для технологии распределенного реестра, субтехнологии не пересекаются и не включают в себя составляющие друг друга. Также субтехнологии не являются отдельными формами применения «сквозной» цифровой технологии [4].

Субтехнологиями систем распределенного реестра являются:

1) технологии организации и синхронизации данных – совокупность методов и инструментов, направленных на определение, организацию и усовершенствование взаимосвязей между частями и

элементами распределенных баз данных, а также на обеспечение их согласованности и приведение к соответствию [4];

2) технологии обеспечения целостности и непротиворечивости данных (консенсус) – совокупность методов и инструментов, направленных на приведение в соответствие имеющихся данных в децентрализованной сети к единой внутренней логике и структуре по заранее определенным правилам, а также обеспечение синхронизации и согласования данных между узлами децентрализованной сети [4];

3) технологии создания и исполнения децентрализованных приложений и смарт-контрактов – совокупность методов и инструментов, направленных на создание приложений, обеспечивающих взаимодействие неограниченного количества участников распределенной системы, и на разработку, поддержание и выполнение компьютерных алгоритмов, предназначенных для автоматизации процессов исполнения контрактов. Децентрализованные приложения обладают прозрачной и открытой логикой, обеспечивающей гарантированное исполнение заданных функций в рамках систем распределенного реестра [4].

Качественные критерии позволяют определить субтехнологию из выборки большого количества технологических решений, признаком которых является то, что с ее помощью создается инфраструктура систем распределенного реестра и определяются правила взаимодействия и проведения обмена данными. Технологии обеспечения целостности и непротиворечивости данных (консенсус) характеризуются содержанием алгоритмов, которые обеспечивают в созданной инфраструктуре неизменность и идентичность данных у участников сети в установленный момент времени. Технологии создания и исполнения децентрализованных приложений и смарт-контрактов обеспечивают возможность использования систем распределенного реестра и включают в свою структуру средства и интерфейсы создания и исполнения смарт-контрактов, обеспечения функциональности децентрализованных приложений и механизмов исполнения алгоритмов и сценариев [4].

2.2. Смарт-контракт как компонент цифровой платформы на базе технологии распределенного реестра

Смарт-контракт отслеживает и обеспечивает исполнение обязательств. Стороны прописывают в нем условия сделки и санкции за их невыполнение, ставят цифровые подписи. Умный контракт самостоятельно определяет, все ли исполнено, и принимает решение: завершить сделку и выдать требуемое (деньги, акции, недвижимость), наложить на участников штраф или пеню, закрыть доступ к активам [5].

Опишем процесс совершения сделки между покупателем и продавцом при помощи смарт-контракта (рисунок 2.1) в рамках платформы по совершению торгово-закупочных операций на агропродовольственном рынке [5].

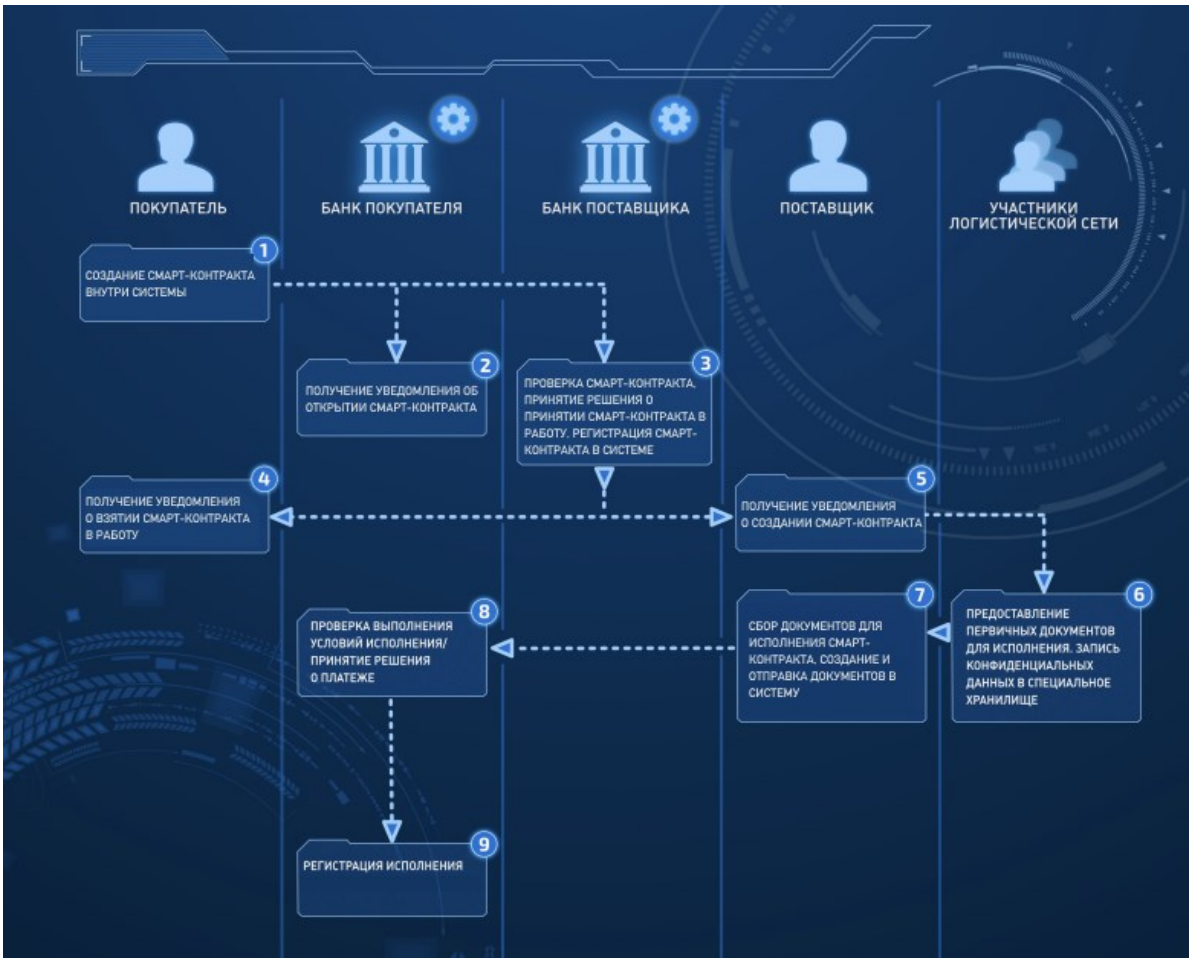


Рисунок 2.1 – Процесс совершения сделки между покупателем и продавцом при помощи смарт-контракта

Описание процесса:

1. В интерфейсе системы покупатель открывает смарт-контракт.
2. Банк покупателя получает уведомление об открытии смарт-контракта, согласовывает условия и размещает в системе информацию о смарт-контракте.
3. Банк поставщика проверяет смарт-контракт, принимает решение о принятии смарт-контракта в работу. Регистрирует смарт-контракт в системе.
4. Покупатель получает уведомление о взятии смарт-контракта в работу.
5. Поставщик получает уведомление о создании смарт-контракта.
6. Логистические компании предоставляют первичные документы для исполнения. Запись конфиденциальных данных в специальное хранилище.
7. Поставщик загружает пакет документов для исполнения смарт-контракта в систему.
8. Смарт-контракт в системе автоматически проверяет выполнение условий исполнения сделки, изменяет статус и инициирует исполнение смарт-контракта банком покупателя.
9. Банк получателя регистрирует исполнение смарт-контракта в системе.

Далее рассмотрим методику создания простейших смарт-контрактов для обеспечения осуществления торгово-закупочных операций цифровой платформы.

2.3. Разбор интерфейса среды разработки смарт-контракта

Смарт-контракт будет разрабатываться под Ethereum, используя язык Solidity. Для разработки на языке Solidity воспользуемся средой Remix. Данная среда Remix работает прямо из браузера и имеет следующий вид:

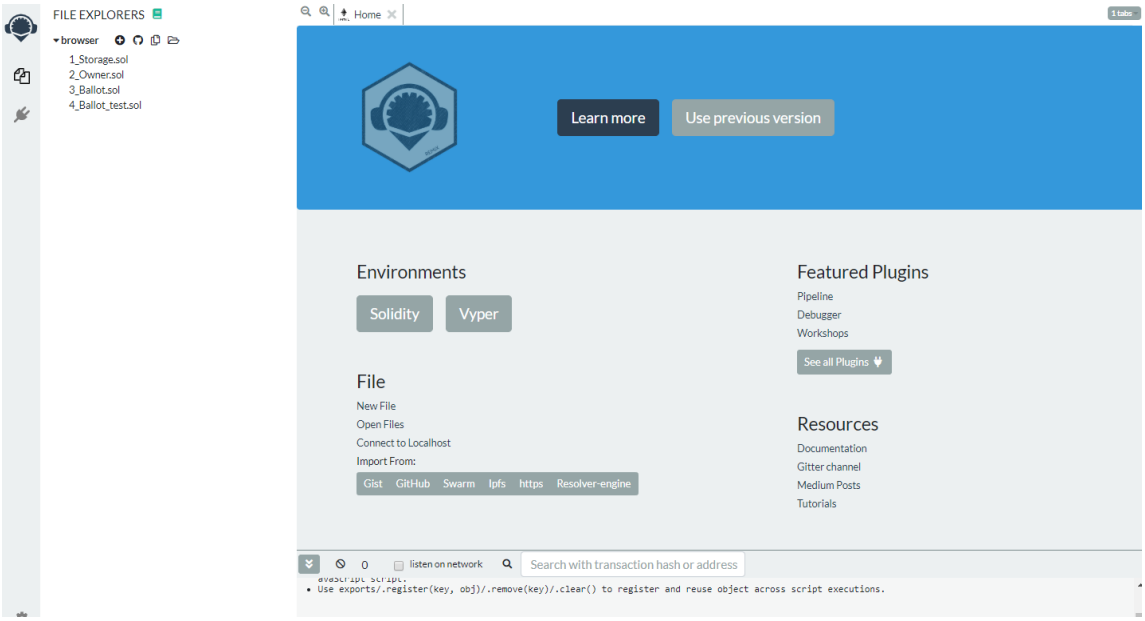


Рисунок 2.2 – Интерфейс программной среды Remix

В левой части расположены инструменты среды, а также список файлов, где можно увидеть несколько контрактов, предлагаемых средой Remix в качестве примеров. Позже в этот список добавим собственные контракты.

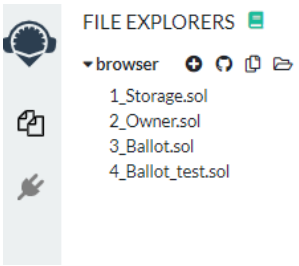


Рисунок 2.3 – Интерфейс программной среды Remix. Инструменты среды и список файлов

Правая часть будет изменяться в зависимости от открытых файлов. Вкладки, позволяющие переключаться между файлами и страницами среды, расположены в самом верху.

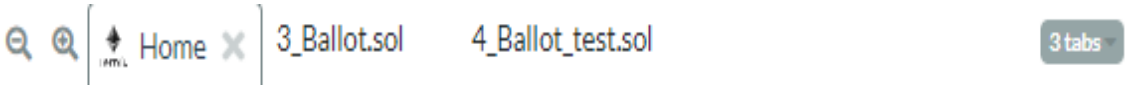


Рисунок 2.4 – Вкладки переключения файлов программной среды Remix

Главная страница, открытая изначально в среде разработки, содержит ссылку на документацию. В нижней части интерфейса расположен выбор среды окружения (Environment), инструмента для работы с тем или иным языком, где необходимо выбрать Solidity. Также на странице расположены кнопки для создания, открытия либо импорта файлов из каких-либо сторонних источников. В правой половине данной страницы можно выбрать необходимые плагины для среды и найти подробную документацию по языку Solidity, а также по работе со средой Remix.

2.4. Первый контракт

Выберем среду в разделе Environments−Solidity и создадим файл HelloWorld.sol. Как понятно из названия, изучение языка мы начнем с вывода «HelloWorld».

На рисунке 2.4 представлен пример кода, который при вызове функции “HelloWorld” вернет “Hello world”.

```
pragma solidity >=0.4.22 <0.7.0;

contract HelloWorld{

    function getHello() public pure returns (string
memory){
        return "Hello world";
    }

}
```

Рисунок 2.4 – Пример кода, который при вызове функции “HelloWorld” вернет “Hello world”

Первая строка показывает, что код написан для Solidity версии 0.4.22, или более новой версии. Сделано это, чтобы не допустить компиляцию данного кода не неподдерживаемой версии языка, где код может работать иначе или не работать вовсе.

Далее необходимо назвать сам контракт. Это чем-то напоминает создание класса в некоторых языках программирования.

Следом описываем функцию getHello, которая является публичной (доступной как извне контракта) чистой (т. е. не взаимодействующей с внутренними переменными контракта) и возвращающей строку в качестве результата. Важно помнить, что при возврате каких-либо значений ссылочного типа (массивы, строки и т. д.), необходимо указывать в каком именно виде будут храниться данные при вызове функции. В данном случае выбран тип memory, который хранит информацию до тех пор, пока выполнение функции не прекратится. Более подробно о некоторых параметрах будет рассказано позже.

В самой функции сделаем непосредственно возврат строки «Hello world».

2.5. Компиляция и проверка

После написания контракта его необходимо скомпилировать и проверить. Для этого нужно открыть инструмент “SOLIDITY COMPILER” на панели инструментов слева, а затем нажать на кнопку «Compile» (рисунок 2.5). В случае возникновения ошибок будет выдано соответствующее сообщение. Если же ошибок не будет, то пользователю предложат узнать дополнительную информацию о скомпилированном контракте, либо же выложить контракт в открытый доступ.

После успешной компиляции необходимо перейти к инструменту “DEPLOY & RUN TRANSACTIONS”, который как ясно из названия позволяет развернуть контракт и запустить транзакции (рисунок 2.6). Для теста воспользуемся окружением JavaScript VM, оставив настройки по умолчанию, чтобы развернуть контракт в тестовой сети, поднятой прямо в браузере. После чего можно нажать на кнопку Deploy и проверить контракт, выполнив функцию getHello.

Если функция объявляется с модификатором pure, то это означает тот факт, что функция не будет взаимодействовать с данными контракта (разрешено только получение констант). Если объявлена функция с модификатором view, то это значит, что функция не будет изменять данные контракта. Модификатор playable позволяет делать платные функции, получающие эфир во время вызова. Помимо этого, для функций можно писать собственные модификаторы.

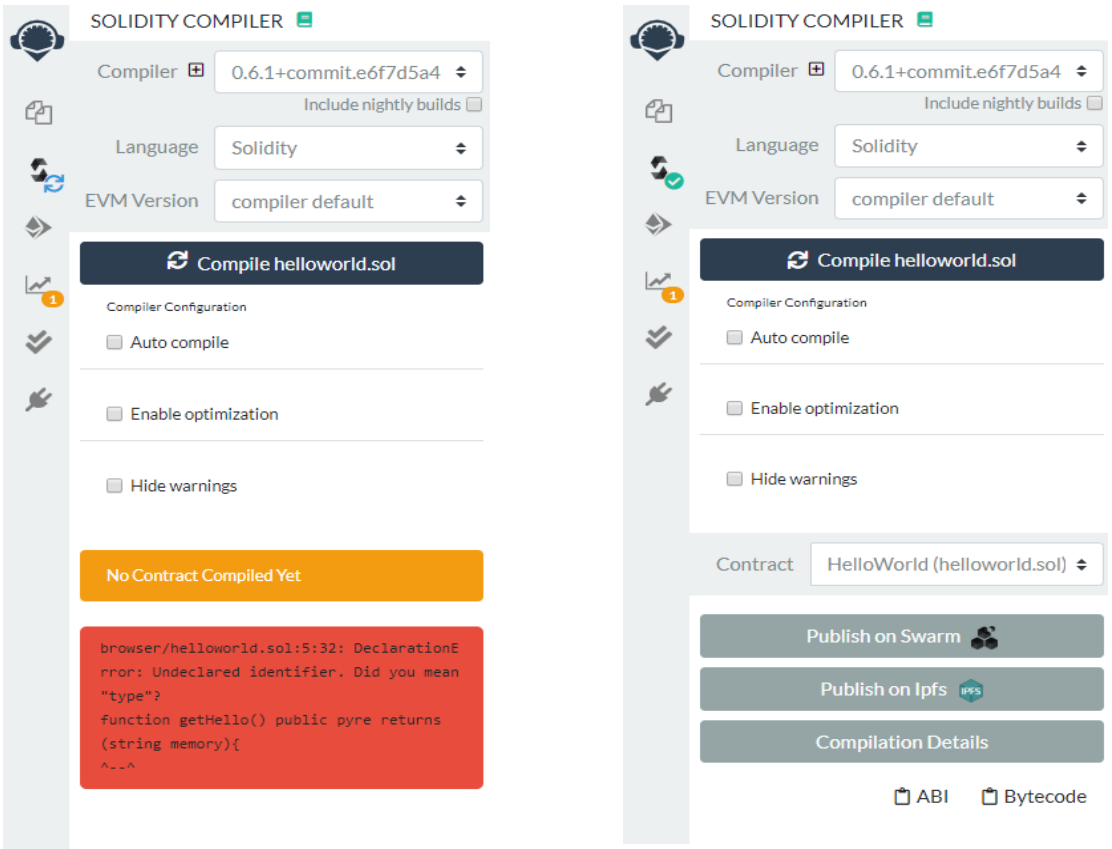


Рисунок 2.5 – Компиляция

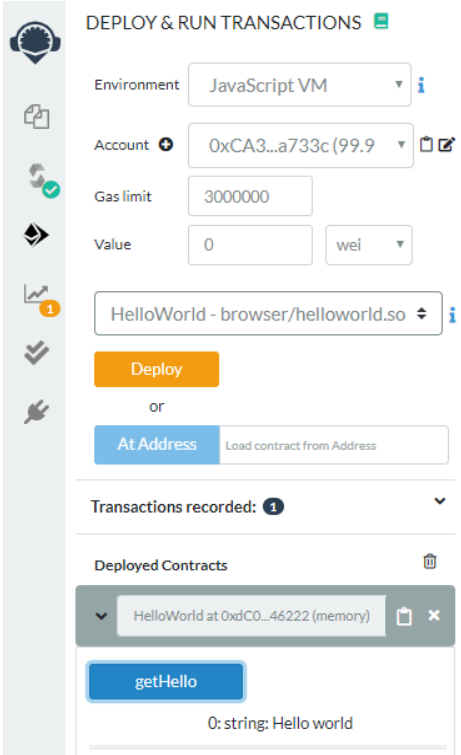


Рисунок 2.6 – Деплой (публикация)

Модификаторы доступа, как понятно из названия, показывают доступность функции. Существуют следующие модификаторы:

external – позволяет вызывать функцию только извне;

public – позволяет вызывать функцию как извне, так и из контракта;

internal – позволяет вызывать функцию из текущего контракта, или контрактов производных(наследуемых) от текущего;

private – позволяет вызывать функцию только из текущего контракта.

Следующий шаг – доработка контракта, чтобы он выводил не строку, созданную прямо в функции, а строку, хранимую в контракте. Для этого создадим в контракте переменную типа string и сразу же присвоим ей значение “Hello world”, затем изменим вывод в функции getHello, чтобы она возвращала именно эту переменную. Но нужно учитывать, что модификаторы функций и модификатор pure в данном случае уже не подойдет, так как происходит взаимодействие с переменной, хранящейся в нашем классе (рисунок 2.7).

```
pragma solidity >=0.4.22 <0.7.0;

contract HelloWorld{

    string text = "Hello world";

    function getHello() public view returns (string
memory){
        return text;
    }

}
```

Рисунок 2.7 – Пример кода

Далее необходимо проверить контракт (не забывайте удалять старые версии контракта из списка) – он также выводит “Hello world”.

Теперь добавим возможность изменения строки прямо из контракта. Для этого нужно написать функцию, принимающую в качестве аргумента новый текст приветствия и записывающую его в пе-

ременную text хранящийся в нашем контракте. Нет необходимости в том, чтобы данная функция что-либо возвращала, а модификаторы pure, или view в данном случае не подойдут, так как функция меняет информацию в контракте (рисунок 2.8).

```
function setHello(string memory newtext) public{
      text = newtext;
}
```

Рисунок 2.8 – Пример кода

Скомпилируем и проверим, что данные действительно изменяются (рисунок 2.9).

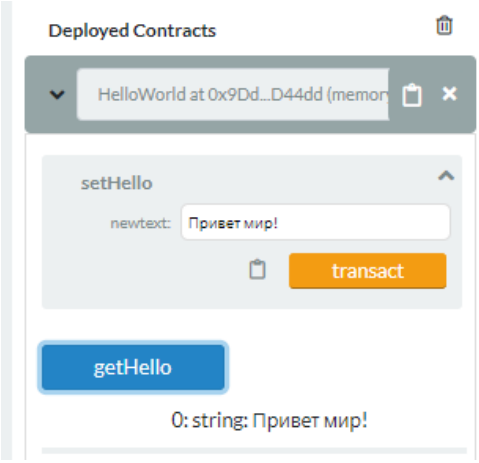


Рисунок 2.9 – Изменение данных

После установки новой строки при помощи функции setHello в getHello должна вернуться установленная строка.

Чтобы описание структуры функций было полным необходимо дополнить его общей схемой написания функции (рисунок 2.10).

```
function [название](аргументы) <модификатор доступа> <модификатор> <возвращаемые значения>{
        [код функции]
 }
```

Рисунок 2.10 – Пример кода

Функции принято называть с маленькой буквы, а контракты – с большой. Функция может принимать и возвращать несколько аргументов, однако, при возврате нескольких аргументов переменные обязательно именовать, а ключевое слово return использовать напротив нет необходимости (рисунок 2.11).

```
function calc(int a, int b) public pure returns(int sum,
int multiply){
        sum = a+b;
        multiply = a*b;
 }
```

Рисунок 2.11 – Пример кода

Переменные описываются как на рисунке 2.12.

```
[тип переменной] <модификатор доступа> <модификатор>
[название переменной] = <значение по умолчанию>;
```

Рисунок 2.12 – Описание переменных

Важно знать, что, если выставить у переменной модификатор доступа public, то извне ее можно будет только получить, но не записать (рисунок 2.13). Сделано это в целях безопасности.

```
string public text = "Hello world";
```

Рисунок 2.13 – Модификатор доступа Public

Помимо модификаторов доступа, переменную можно сделать константой (рисунок 2.14).

```
string public constant text = "Hello world";
```

Рисунок 2.14 – Переменная константа

2.6. Владельцы контрактов, понятие mapping, собственные модификаторы

2.6.1. Владельцы контрактов

В прошлом разделе была приведена методика написания простого смарт-контракта, которая позволяет выводить приветствие и его менять. Но на данный момент менять приветствие может любой пользователь, знающий адрес контракта. Для того чтобы изменять приветствие мог только создатель контракта, необходимо доработать код, как на рисунке 2.15.

```
pragma solidity >=0.4.22 <0.7.0;

contract HelloWorld{

    string text = "Hello world";
    address owner;

    constructor() public{
        owner=msg.sender;
    }

    function getHello() public view returns (string
memory){
        return text;
    }

    function setHello(string memory newtext) public{
        require(owner==msg.sender, "you not owner!");
        text = newtext;
    }
}
```

Рисунок 2.15 – Пример кода

address − 20 битный адрес, используемый для хранения адресов (пользователей и контрактов) сети Ethereum.

msg.sender − содержит адрес пользователя сети Ethereum, вы-

звавшего данную функцию.

constructor − как понятно из названия − функция, вызываемая при публикации (deploy) контракта, если данная функция принимает аргументы − эти аргументы будут необходимы, при публикации контракта.

require − прерывает выполнение операции, если условие в первом аргументе ложно. Допускает использование второго аргумента для указания текста ошибки.

В переменной owner хранится адрес пользователя сети Ethereum, его можно проверить в функции изменения текста. Если пользователь, пытающийся изменить текст сообщения, не является создателем контракта, то будет выдано соответствующее сообщение (рисунок 2.16).

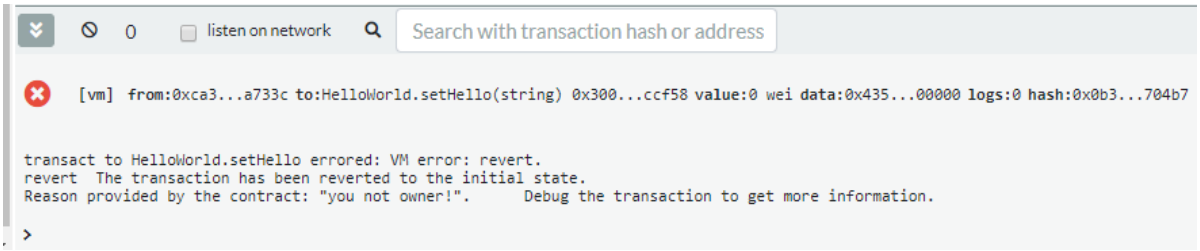


Рисунок 2.16 – Пример ошибки

Чтобы проверить контракт на другом пользователе, можно в инструменте публикации и запуска выбрать необходимый аккаунт виртуальной сети, или же добавить новый.

Важно помнить, что опубликованный контракт нельзя изменить, его можно только опубликовать заново (в реальной сети удаление контрактов также невозможно).

2.6.2. Mapping

Порой может возникнуть необходимость в нескольких владельцах. Конечно, их можно хранить в массиве и каждый раз перебирать весь массив в поисках нужного, но это не совсем рационально и создает нагрузку при обработке. Куда более рациональным вариантов будет хранение адреса пользователя и логической переменной в Mapping.

Mapping представляет собой некий структурированный массив, где вместо цифровых индексов можно применять любой стандартный тип данных (байт, строка, адрес и даже enum). Другие типы, созданные пользователем, или сложные типы не допускаются. В качестве значения можно использовать любой тип данных.

В качестве примера использования Mapping необходимо сделать отдельную функцию, которая будет приветствовать пользователя его собственным текстом. Для этого нужно определить mapping usertext с ключом равным адресом пользователя и хранящим текст приветствия. Затем добавить выставление соответствующих значений и вывод приветствия из usertext (рисунок 2.17).

```
    mapping(address => string) usertext;

    function getUserHello() public view returns (string
memory){
        return usertext[msg.sender];
    }

    function setUserHello(string memory newusertext) pub-
lic{
        usertext[msg.sender]=newusertext;
    }
```

Рисунок 2.17 – Пример кода

В завершении темы mapping`ов важно сделать так, чтобы несколько пользователей одновременно могли быть владельцами данного контракта. Для этого нужно создать похожий mapping, только вместо строк будем в нем хранить логическую переменную типа bool. При изменении глобального текста приветствия проверяется, что данный пользователь является администратором. Также необходимо ввести функцию добавления и соответственно удаления новых администраторов путем ввода адреса пользователя нынешним администратором.

Часть кода, отвечающая за вышеперечисленное, будет выглядеть так, как представлено на рисунке 2.18.

```
    mapping(address => bool) owners;

    constructor() public{
        owners[msg.sender]=true;
    }

    function getHello() public view returns (string
memory){
        return text;
    }

    function setHello(string memory newtext) public{
        require(owners[msg.sender], "you not owner!");
        text = newtext;
    }

    function addOwner(address newowner) public{
        require(owners[msg.sender], "you not owner!");
        owners[newowner]=true;
    }

    function remOwner(address remowner) public{
        require(owners[msg.sender], "you not owner!");
        owners[remowner]=false;
    }
```

Рисунок 2.18 – Пример кода

2.6.3. Кастомные модификаторы

Как можно заметить, довольно часто приходится делать одинаковые проверки на то, что пользователь является администратором. Но подобные проверки, и не только проверки можно вынести в отдельный модификатор. Модификаторы имеют вид, представленный на рисунке 2.19.

```
modifier [название модификатора](<аргументы модификатора>)
{
        [действия выполняемые до выполнения основного кода
функции]
```

```
        _;//Здесь будет вставлено тело основной функции
        [действия выполняемые по-
сле выполнения основного кода
функции]
}
```

Рисунок 2.19 – Пример кода

Модификаторы позволяют дополнять основной код функции, а оператор “_;” вызывает функцию, к которой применен данный модификатор.

Возвращаясь к конкретному примеру, напишем модификатор проверки, что пользователь является владельцем контракта. Выглядеть он будет как на рисунке 2.20.

```
modifier onlyOwner() {
        require(owners[msg.sender], "you not owner!");
        _;
}
```

Рисунок 2.20 – Пример кода

Для полноты картины необходимо написать модификатор, проверяющий, что пользователь, которого удаляют из владельцев, не является отправителем данной команды (не пытается удалить сам себя). Для этого нам необходимо в аргументах модификатора принять адрес пользователя, которого мы хотим удалить из администраторов и сверить его с автором сообщения. Выглядеть это будет примерно так, как на рисунке 2.21.

```
modifier removeSelf(address remowneradress) {
        require(remowneradress!=msg.sender, "you can't de-
lete yourself!");
        _;
}
```

Рисунок 2.21 – Пример кода

Теперь нужно добавить модификаторы к функциям. Модификаторы без аргументов можно вызывать как со скобками, так и без (рисунок 2.22).

```
    function setHello(string memory newtext) public on-
lyOwner(){
        text = newtext;
    }

    function addOwner(address newowner) public onlyOwner{
        owners[newowner]=true;
    }

    function remOwner(address remowner) public onlyOwner
removeSelf(remowner){
        owners[remowner]=false;
    }
```

Рисунок 2.22 – Пример кода

С модификаторами код становится менее нагруженным и более читабельным. bool может содержать true, или false. Код, описанный ниже (рисунок 2.23), установит btest на true (утверждение, что *a* не равно *b* – истина).

```
int a=5;
int b=6;
bool btest = a!=b;
```

Рисунок 2.23 – Пример кода

Int/Uint – целые числа со знаком (Int) либо без знака (unsigned) (Uint).

Позволяет задавать размер числа в битах от 8 до 256 с шагом 8 сразу в указании типа (int8/uint8 – int256-uint256). При указании типа int/uint без размера создается число с размером 256 бит (int256 или uint256 соответственно).

Fixed/Ufixed – знаковые либо беззнаковые числа с фиксированной точкой. Задаются посредством ключевого слова fixedMxN,

либо UfixedMxN, где M – размер числа в битах, а N – количество знаков после точки(запятой).

Address – 20-битный адрес, используемый для хранения адресов сети Ethereum. Существует две разновидности адреса:

Address − просто содержит адрес пользователя;

Address playable − тот же адрес, но содержащий дополнительные методы, такие как transfer и send.

Основные операции, которые будут производить с адресом:

1. Сравнение двух адресов (==).
2. Узнать баланс по адресу (<address>.balance).
3. Сделать перевод на адресс (<address payable>.transfer).

Байтовые массивы фиксированного размера – значения типа bytes1, bytes2, bytes3 … bytes32 позволяют хранить последовательность байт длиной от 1 до соответствующего размера массива.

Иногда возникает необходимость описать несколько состояний одного контракта, которые будут всегда именно такими и не будут меняться. К примеру нужно описать статус заказа not_started,in_progress, ready_to_issue (работа не начата, в работе, готов к выдаче). Использовать строки для хранения подобных данных будет неудобно, да и не совсем рационально. Использование числа будет куда рациональней, но также неудобно для программиста. Именно в данном случае используется enum. Он позволяет перечислить необходимые состояния и работать с ними, в то же время данные представляются в виде цифровых значений, что позволяет избежать лишнего использования памяти (рисунок 2.24).

```
enum contractState { not_started, in_progress,
ready_to_issue }
```

Рисунок 2.24 – Пример enum

Элемент enum`a можно явно преобразовать в int, либо uint. Также наоборот – uint и int можно явным образом преобразовать в enum.

Для примера напишем небольшой контракт, хранящий текущий статус контракта, позволяющий его вывести и изменить как явным указанием кода контракта, так и изменив его на следующий.

А также сделаем вывод текущего состояния контракта в виде текста (рисунок 2.25).

```
pragma solidity >=0.4.22 <0.7.0;

contract enums{
    enum contractState { not_started, in_progress,
ready_to_issue }
    contractState state;

    function getState() public view returns (con-
tractState){
        return state;
    }

    function getStateStr() public view returns (string
memory){
        if(state==contractState.not_started){
            return "Sorry, the contract hasn't started
yet";
        }
        else if(state==contractState.in_progress){
            return "sorry, the contract is in the works";
        }
        else if(state==contractState.ready_to_issue){
            return "The contract is ready. You can pick up
product!";
        }
    }

    function stState(contractState _state) public{
        state = _state;
    }

    function nextstate() public{
        state = contractState(uint(state)+1);
    }
}
```

Рисунок 2.25 – Пример кода

Enum хранится в виде числовой переменной внутри контракта, однако работать с ним куда удобней, нежели с числовой переменной.

2.7. Платежные операции

2.7.1. Секретный предмет

Для реализации товара необходимо выбрать предмет, который будем продавать. В этой роли выступит особый контракт, содержащий секретную строку, которую может задавать и смотреть только владелец, также добавим возможность смены владельца (рисунок 2.26).

```
pragma solidity >=0.4.22 <0.7.0;

contract secret{
    string private secrettext;
    address public owner;

    constructor() public{
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(owner==msg.sender, "you not owner!");
        _;
    }

    function getSecret() public view onlyOwner re-
turns(string memory){
        return secrettext;
    }

    function setSecret(string memory _secrettest) public
onlyOwner{
        secrettext = _secrettest;
    }

    function changeOwner(address _owner) public onlyOwner{
```

```
        owner = _owner;
    }

}
```

Рисунок 2.26 – Пример кода

2.7.2. Статусы сделки

Предположим, что пользователь развернул данный контракт и сохранил в него какую-то секретную информацию, а теперь хочет продать ее. При том продать данную информацию хочется не просто, а при помощи безопасной сделки.

Контракт продажи будет иметь один из 4 различных статусов сделки:

- notstarted (не начата);
- started (начата: ожидает покупателя);
- canceled (отменена);
- sucessfully (успешно выполнена).

Подобные значения удобно хранить как enum.

Также необходимы адреса покупателя и продавца, адреса именно платежные, ведь мы будем работать с деньгами (хоть и в тестовой сети). Безусловно, необходимо хранить адрес продаваемого предмета, притом будет полезно его сразу структурировать, чтобы иметь возможность обращаться к его функциям. И, конечно, необходима цена продажи. Любой пользователь может увидеть адрес продаваемого предмета, а также цену его продажи. При создании сделка должна принимать предмет продажи и цену, сохранять создателя сделки как получателя.

После создания необходимо, чтобы пользователь передал права на секретный предмет нашему контракту, а в случае отмены контракта имел возможность его вернуть. Отменить контракт может только продавец и то до тех пор, пока не был найден покупатель. При отмене статус сделки меняется на canceled.

Начать продажу (поиск покупателя) может только продавец, в том случае, если права на секретный предмет уже принадлежат сделке. После начала сделки статус сделки меняется на started. Купить предмет возможно только, если сделка находится в состоянии

started. Продажа пройдет успешно только в том случае, если сумма, отправленная покупателем, больше либо равна сумме продажи. Если покупатель отправил больше, чем необходимо, то система вернет лишнюю сумму обратно владельцу.

После покупки покупатель должен иметь возможность забрать свой товар (стать владельцем товара), а продавец деньги, полученные от продажи товара.

Для реализации покупки большое значение имеет команда import и функции, связанные с оплатой. Команда import [filename] позволяет обращаться к контрактам, содержащимся в другом файле. Допустимо использовать ключевого слово as для указания переменной, через которую будут происходить обращения именно к этому файлу (рисунок 2.27).

```
import [filename] as sympolname
```

Рисунок 2.27 – Пример кода

Модификатор:

payable позволяет использовать в функции методы, связанные с платежными переводами;

[address].transfer ([сумма перевода]) позволяет сделать перевод указанной суммы на указанный адрес;

[address].balance позволяет узнать баланс пользователя, либо контракта.

Платежные средства могут быть не только у пользователей, но и у контрактов (рисунок 2.28).

```
pragma solidity >=0.4.22 <0.7.0;
import "./secret.sol";
contract secret_sell{
    enum sell_state{notstarted, started, canceled, success-
fully}
    sell_state public state;
    address payable seller;
    address payable buyer;
    secret public item;
    uint public amount;
```

```
    modifier isState(sell_state _state){
        require(state==_state, "state is not valid!"); _;
    }
    modifier isItemOwner(){
        require(item.owner()==address(this), "sell is not
owner of secret!"); _;
    }
    modifier isSellNotEnd(){
        re-
quire(state==sell_state.notstarted||state==sell_state.start
ed, "sell is ended!"); _;
    }
    modifier isSeller(){
        require(msg.sender == seller, "you not seller!");
_;
    }
    modifier isBuyer(){
        require(msg.sender == buyer, "you not buyer!"); _;
    }
    modifier cost(){
        require(msg.value >=amount);
        buyer = msg.sender;
        _;
        if(msg.value>amount){
            buyer.transfer(msg.value-amount);
        }
    }
    constructor(secret _item, uint _amount) public{
        seller = msg.sender;
        item = _item;
        amount = _amount;
    }
    function setSellerItemOwner() public is-
State(sell_state.canceled) isSeller() isItemOwner(){
        item.changeOwner(seller);
    }
    function GetSelMoney() public isSeller() is-
State(sell_state.successfully){
        seller.transfer(address(this).balance);
    }
    function setBuyerItemOwner() public is-
```

```
State(sell_state.successfully) isBuyer() isItemOwner() {
        item.changeOwner(buyer);
    }
    function cancel() public isSellNotEnd isSeller{
        state = sell_state.canceled;
    }
    function startSell() public isItemOwner is-
State(sell_state.notstarted) isSeller{
        state = sell_state.started;
    }
    function buy() public payable is-
State(sell_state.started) cost(){
        state = sell_state.successfully;
    }
}
```

Рисунок 2.28 – Пример кода

Так будет выглядеть простейшая реализация описанного выше функционала. Msg.value: сумма, которую отправил пользователь при вызове функции. К слову, в тестовой среде она указывается в значении value.

Теперь для закрепления попробуйте самостоятельно реализовать не продажу секрета, а обмен секретами с другим пользователем. И конечно сразу нужно понимать минусы подобной реализации продажи – необходимость от пользователя лишних действий, таких как передача прав на секрет контракту продажи. Однако у такого подхода есть и плюсы: отсутствие необходимости предусматривать продажу при разработке контракта секрета.

2.8. Наследование и работа со временем

2.8.1. Наследование

Чтобы владельцем контракта был тот или иной пользователь сети, будем использовать переменную owner, которую выставляем в конструкторе, а также модификатор isOnlyOwner, который позволяет проверить права администратора (рисунок 2.29).

```
contract Secret{
    string private secrettext;
    address public owner;
    constructor() public{
        owner = msg.sender;
    }
    modifier isOnlyOwner() {
        require(owner==msg.sender, "you not owner!");
        _;
    }

    function getSecret() public view isOnlyOwner re-
turns(string memory){
        return secrettext;
    }

    function setSecret(string memory _secrettest) public
isOnlyOwner{
        secrettext = _secrettest;
    }
}
```

Рисунок 2.29 – Пример кода

Безусловно, писать подобное в каждом контракте, где необходим владелец не всегда удобно. Поэтому вынесем эту часть кода в отдельный контракт (рисунок 2.30).

```
contract Owned{
    address public owner;

    constructor() public{
        owner = msg.sender;
    }

    modifier isOnlyOwner() {
        require(owner==msg.sender, "you not owner!");
        _;
    }
}
```

Рисунок 2.30 – Пример кода

А в класс Secret просто унаследуем от контракта Owned при помощи ключевого слова is (рисунок 2.31).

```
contract Secret is Owned{
    string private secrettext;

    function getSecret() public view isOnlyOwner re-
turns(string memory){
        return secrettext;
    }

    function setSecret(string memory _secrettest) public
isOnlyOwner{
        secrettext = _secrettest;
    }
}
```

Рисунок 2.31 – Пример кода

Если развернуть контракт, увидим, что все публичные методы и переменные наследуемого контракта доступны и в нашем контракте.

2.8.2. Переопределение функций

Немаловажно, что можно наследовать сразу несколько контрактов и переопределять те функции, которые необходимы (если наследуемый класс это позволяет).

Для теста опишем простейший контракт, который будет содержать две функции getText, доступные в рамках контракта и в наследуемых контрактах, а также функцию ShowText, которая выводит текст, полученный из данной функции (рисунок 2.32).

```
contract Text{
    function getText() internal pure virtual returns(string
memory){
        return "Object";
    }

    function ShowText() public pure returns (string
```

```
memory){
        return getText();
    }
}
```

Рисунок 2.32 – Пример кода

Функция getText содержит модификатор virtual. Данный модификатор показывает, что данную функцию можно перезагрузить при наследовании. Если просто унаследоваться от данного класса, то при вызове функции ShowText будет выдаваться строка “Object”, но можно переопределить это поведение. Для этого создадим функцию getText в контракте Secret с модификатором override, который позволяет перезагружать методы наследуемых классов (рисунок 2.33).

```
contract Secret is Ownered, Text{
    string private secrettext;

    function getSecret() public view isOnlyOwner re-
turns(string memory){
        return secrettext;
    }

    function setSecret(string memory _secrettest) public
isOnlyOwner{
        secrettext = _secrettest;
    }

    function getText() internal pure override re-
turns(string memory){
        return "Secret object";
    }
}
```

Рисунок 2.33 – Пример кода

Теперь, если вызвать функцию ShowText, вернется сообщение с текстом “Secret object”. Чтобы получить текущее время, можно

воспользоваться оператором now, который вернет текущее время в секундах в виде неотрицательного целого числа. Далеко не всегда удобно использовать данное число для расчета текущей даты, однако от данного числа можно откладывать какое-либо время. Например, можно указать, сколько времени будет действовать предложение продажи. Однако далеко не всегда удобно оперировать такими мелкими единицами как секунды. Поэтому в solidity предусмотрены следующие глобальные константы:

1 minutes − 60 с

1 hours − 60 мин

1 days − 24 ч

1 weaks − 7 дней

Так как речь зашла про единицы времени, нельзя не упомянуть и единицы эфира. Изначально, при работе с эфиром имеем дело с wei, что представляет из себя 1 квинтилионную (18 нулей) часть эфира. Далее будет представлена небольшой список единиц эфира.

1 wei = 1

1 szabo = 1e12 (12 нулей)

1 finney = 1e15

1 ether = 1e18

Все мелкие единицы эфира названы в часть первопроходцев в области криптографии, например wei названа в часть Wei Dai , азиатского криптографа.

Для создания контрактов внутри существующих можно использовать ключевое слово new, однако нужно помнить, что на момент компиляции текущего контракта создаваемый в процессе работы контракт уже должен быть скомпилирован (рисунок 2.34).

```
Trader traderT = new Trader(amount*1 ether);
```

Рисунок 2.34 – Пример кода

Доработайте контракт для более удобной работы пользователя с ним. Первое, что нужно учитывать, это чтобы при выставлении на продажу контракта не терялся контроль над ним и оставался владелец. Второе, что должна быть реализована простая функция,

которая сама создает сделку продажи и дает ей необходимые права.

Для реализации вышеперечисленного необходимо создать контракт Sellable для объектов, которые будет продаваться. В свою очередь отнаследуем от контракта owner, чтобы передать права владельца покупателю, в случае успешной сделки. Для этого нужна переменная, куда сможем записывать адрес контракта, отвечающего на данный момент за продажу, если адрес равен нулю, то объект не продается. Помимо этого, нужна функция выставления на продажу (создания нашего контракта продажи и записи его в качестве ответственного за продажу), также нужны функции, доступные для вызова только контракта, отвечающего за продажу – передача прав на объект покупателю и завершение продажи (снятие с продажи в случае успешной продажи или ее отмены) (рисунок 2.35).

```
contract Sellable is Owned{
    address public trader;

    modifier isOnlyTrader() {
        require(trader==msg.sender, "you not trader!");
        _;
    }

    modifier isNotSelling(){
        require(trader==address(0), "object alredy sell-
ing!");
        _;
    }

    function sell(uint amount) public isOnlyOwner isNot-
Selling{
        require(amount>0, "the amount must be greater than
0!");
        Trader traderT = new Trader(amount*1 ether);
        trader = address(traderT);
    }

    function changeOwnerToByuer(address byuer) external
isOnlyTrader{
        owner = byuer;
    }
```

```
    function endSell() external isOnlyTrader{
        trader=address(0);
    }
}
```

Рисунок 2.35 – Пример кода

Теперь необходимо унаследовать контракт Secret от контракта Sellable и описать контракт Trader. В данной реализации он будет принимать в конструктор только сумму продажи, все остальные необходимые ему данные он будет получать из msg.sender приведенной к типу Sellable. В нашем контракте Trader необходимо хранить:

Адрес продаваемого предмета.

Продавца. (платежный адрес)

Покупателя. (платежный адрес)

Сумму продажи.

Статус продажи. (к которому был добавлен статус locket, когда контракт ожидает подтверждения сделки со стороны покупателя).

Время старта продажи.

Максимальная продолжительность продажи.

Пример реализации конструктор контракта Trader представлен на рисунке 2.36.

```
   constructor (uint _amount) public{
        sellItem = Sellable(msg.sender);
        seller = payable(sellItem.owner());
        amount = _amount;
    }
```

Рисунок 2.36 – Пример кода

Следующее, что необходимо реализовать – это функция отмена. Продавец может отменить продажу в случае, если покупатель еще не был найден, либо в случае, если контракт ожидает подтверждения со стороны покупателя, но уже вышло отведенное время. Покупатель же может отменить продажу только когда контракт

находится в ожидании подтверждения сделки со стороны покупателя. В случае отмены сделки необходимо установить контракту соответствующий статус, вернуть ранее внесенные средства покупателю (если они уже были внесены), а также снять продаваемый объект с продажи.

При старте продаж необходимо записать текущее время и принять в качестве аргументов максимальную продолжительность продажи (время, когда продажа будет активна). И установить статус начала продажи.

Функция покупки (внесения средств за товар) предполагает проверку количества внесенных средств пользователем и регистрацию пользователя, внесшего средства в качестве покупателя. Помимо этого, необходимо изменить статус контракта на заблокированный (ожидающий подтверждения). В случае, если пользователь внес больше, чем необходимо, нужно вернуть лишние средства обратно пользователю. Функция доступна только если сделка находится в состоянии started и время продажи еще не истекло.

Функция подтверждения доступна только покупателю и только на этапе подтверждения покупки. В случае подтверждения устанавливается статус сделки на sucesfull, меняется владелец продаваемого предмета на покупателя и передаются деньги, находящиеся на контракте (полученные от покупателя) продавцу, а также снимается продаваемый предмет с продажи.

Реализация данного контракта отражена на рисунке 2.37.

```
contract Trader{
    Sellable public sellItem;
    address payable seller;
    address payable buyer;
    uint public amount;
    enum E_TradeState{notstaeted, started, locked, can-
celed, sucessfull}
    E_TradeState public state;
    uint TradeStartTime;
    uint TradeDuration;

    modifier isState(E_TradeState _state){
        require(state==_state, "not valid state!");
        _;
```

```
    }

    modifier isSeller(){
        require(msg.sender==seller, "You not seller!");
        _;
    }

    modifier isBuyer(){
        require(msg.sender==buyer, "You not buyer!");
        _;
    }

    modifier isTimeNotEnd(){
        require(TradeStartTime+TradeDuration>now, "Sorry,
the sale time has expired.");
        _;
    }

    constructor (uint _amount) public{
        sellItem = Sellable(msg.sender);
        seller = payable(sellItem.owner());
        amount = _amount;
    }

    function start(uint _TradeDurationMinutes) public is-
Seller isState(E_TradeState.notstaeted){
        require(sellItem.trader()==address(this));
        TradeStartTime = now;
        TradeDuration = _TradeDurationMinutes * 1 minutes;
        state = E_TradeState.started;
    }

    function addDuration(uint _addTradeDurationMinutes)
public isSeller isState(E_TradeState.started){
        TradeDuration+=_addTradeDurationMinutes * 1
minutes;
    }

    function getTimeToEndSell() public view returns(uint){
        if(TradeStartTime+TradeDuration>now){
            return TradeStartTime+TradeDuration-now;
        }else{
```

```
            return 0;
        }
    }

    function isCancelSeller()private view returns(bool){
        return state<=E_TradeState.started ||
(state==E_TradeState.locked && now > TradeStartTime +
TradeDuration);
    }

    function cancelProcess() private{
        state = E_TradeState.canceled;
        if(buyer!=address(0)){
            buyer.transfer(address(this).balance);
        }
        sellItem.endSell();
    }

    function cancel() public{
        if(msg.sender==seller && isCancelSeller()){
            cancelProcess();
        }else if(msg.sender == buyer &&
state==E_TradeState.locked){
            cancelProcess();
        }else{
            revert("You do not have permission for this ac-
tion!");
        }
    }

    function confirm() public isBuyer is-
State(E_TradeState.locked){
        state = E_TradeState.sucessfull;
        sellItem.changeOwnerToByuer(buyer);
        seller.transfer(address(this).balance);
        sellItem.endSell();
    }

    function buy() public payable is-
State(E_TradeState.started) isTimeNotEnd(){
        require(msg.value >= amount, "not enough money!");
```

```
        state = E_TradeState.locked;
        buyer = msg.sender;

        if(msg.value>amount){
            buyer.transfer(address(this).balance-amount);
        }
    }
}
```

Рисунок 2.37 – Пример кода

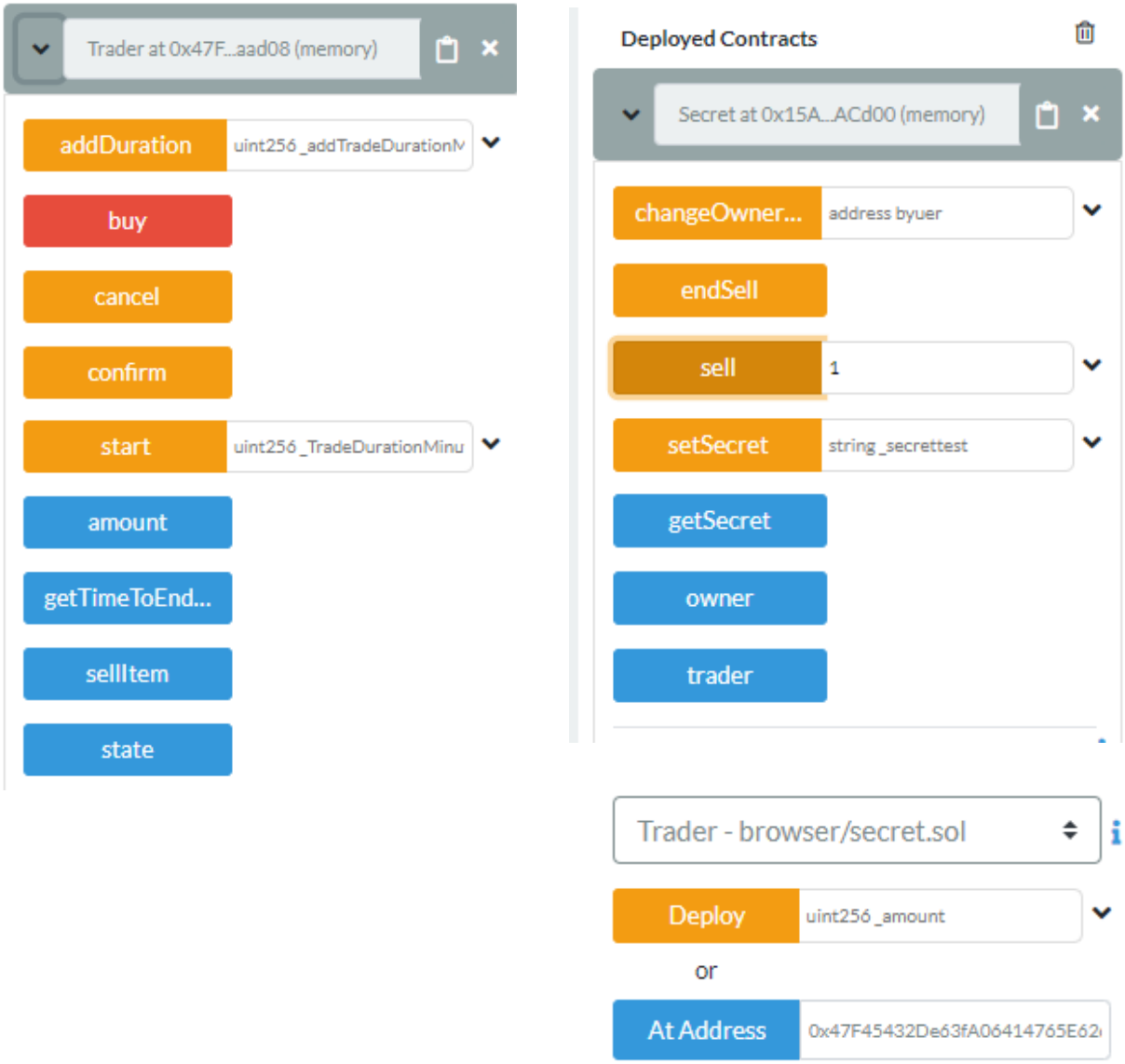


Рисунок 2.38 – Компиляция и развертка контракта

Чтобы проверить контракт, необходимо скомпилировать и развернуть секретный контракт, записать в него секретный текст и продать его за 1 ether. После чего создастся контракт Trader. Чтобы увидеть его, нужно нажать получить адрес trader`a через соответствующую кнопку, а затем обратиться к нему, выбрав соответствующий контракт и воспользовавшись кнопкой At Address. Затем можно начать продажу и купить товар другим пользователем виртуальной сети и убедиться в том, что секрет действительно перешел во владение другому пользователю (рисунок 2.38).

2.9. Массивы, циклы, события

2.9.1. Массивы

Массив – пронумерованная последовательность значений одного типа, обозначаемая одним именем. Массив можно задать, посредством написания типа данных и квадратных скобок после него, а затем названия массива (рисунок 2.39).

```
ТипДанных[] Название;
```

Рисунок 2.39 – Пример кода

Например, массив типа Int будет задан так, как изображено на рисунке 2.40.

```
Int[] nums;
```

Рисунок 2.40 – Пример кода

Добавлять элементы в массив можно посредством команды push() (рисунок 2.41).

```
nums.push(n);
```

Рисунок 2.41 – Пример кода

Обращаться к элементам массива можно по индексу, указанному в квадратных скобках после имени массива (рисунок 2.42).

```
nums[index];
```

Рисунок 2.42 – Пример кода

Создадим контракт, который будет содержать массив и команды для работы с ним. Добавление нового элемента, вывод элемента по индексу и вывод всего массива представлен на рисунке 2.43.

```
contract array{
    int[] nums;
    function getNumsCount()public view returns(uint){
        return nums.length;
    }

    function addNum(int n) public{
        nums.push(n);
    }

    function listNums() public view returns(int[] memory){
        return nums;
    }

    function geNum(uint index) public view returns(int){
        return nums[index];
    }
}
```

Рисунок 2.43 – Пример кода

2.9.2. Циклы и события

В Solidity существует три вида циклов:
for − цикл со счетчиком;
while - цикл с предусловием;
do while − цикл с постусловием.

```
for([до начала цикла];[условие цикла];[каждый шаг цикла]){
     //Тело цикла
}
```

Рисунок 2.44 – Пример кода

Типовое описание цикла for. Пример цикла for представлен на рисунке 2.45.

```
for(int i=0; i<10; i++){
     //Тело цикла
}
```

Рисунок 2.45 – Пример кода

Такой цикл сначала создаст переменную i и установит ее значение на 0, затем проверит соблюдение условия, что i < 10 и в конце каждого шага будет прибавлять к i единицу.

Типовое описание цикла while отображено на рисунке 2.46.

```
while([условие]){
     //Тело цикла
}
```

Рисунок 2.46 – Пример кода

Типовое описание цикла do while представлено ниже (рисунок 2.47).

```
do{

}while([условие]);
```

Рисунок 2.47 – Пример кода

Разница между while и do while лишь в одном, что while не начнет выполнение цикла, если условие не соблюдается, тогда как do while выполнит один шаг цикла, но прекратит дальнейшее вы-

полнение при несоблюдении условия.

При желании цикл можно прервать в любой момент посредством оператора break.

Добавим в контракт функцию, считающую сумму элементов массива (рисунок 2.48).

```
function sumNums() public view returns(int){
        int sum=0;
        for(uint i=0; i<nums.length; i++){
            sum+=nums[i];
        }
        return sum;
 }
```

Рисунок 2.48 – Пример кода

События задаются ключевым словом Event, после которого следует название события, а в скобках описываются аргументы события (рисунок 2.49).

```
    event НазваниеСобытия (АргументыСобытия);
```

Рисунок 2.49 – Пример кода

Event можно вызвать из любой точки функции контракта посредством ключевого слова emit (рисунок 2.50).

```
    emit НазваниеСобытия(ЗначенияАргументов);
```

Рисунок 2.50 – Пример кода

События нужны для взаимодействия с интерфейсом контракта и передачи ему данных о статусе выполнения операции. Например, для описания визитки, содержащей ФИО, должность и телефон сотрудника необходимо создать следующую структуру (рисунки 2.51, 2.52).

```
struct BuisnessCard{
    string FIO;
    string Post;
    uint phone;
}
```

Рисунок 2.51 – Пример кода

Ниже представлен пример использования данной структуры.

```
struct BuisnessCard{
    string FIO;
    string Post;
    uint phone;
}
```

Рисунок 2.52 – Пример кода

Таким образом, задача по созданию смарт-контракта была декомпозирована, были разработаны вышеперечисленные блоки, которые позволяют автоматизировать процесс заключения договоров купли-продажи для цифровой агропродовольственной платформы, повысить прослеживаемость бизнес-процесса продажи продукции для всех участников торгово-закупочных операций, а также для надзорных органов власти.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. **Водянников В. Т.** Этапы совершенствования технических средств и тенденции сменяемости технологических укладов экономики // Экономика сельского хозяйства России. 2022. № 3. С. 17−21.
2. Гарант.ру [Электронный ресурс]. URL: https://www.garant.ru.
3. Постановление Правительства РФ от 14 ноября 2015 г. № 1235 «О федеральной государственной информационной системе координации информатизации» [Электронный ресурс]. URL: http://pravo.gov.ru/proxy/ips/?docbody=&prevDoc=102132393&backlink=1&&nd=102382687.
4. Дорожная карта развития «сквозной» цифровой технологии «Системы распределенного реестра» [Электронный ресурс]. URL: https://rfrit.ru/media/documents/%D0%94%D0%9A_%D0%A1%D0%A6%D0%A2_%D0%A2%D0%A0%D0%A0.pdf
5. Как можно оптимально использовать технологию смарт-контрактов при создании бизнеса [Электронный ресурс]. URL: https://uspet-vse.ru/articles/kak-mozhno-optimalno-ispolzovat-tehnologiyu-smart-kontraktov-pri-sozdanii-biznesa.html.
6. **Козубенко И. С.** Почвенная информация в аналитическом центре минсельхоза России // Бюллетень Почвенного института им. В. В. Докучаева. 2018. № 92. С. 3−15. DOI 10.19047/0136-1694-2018-92-3-15.
7. **Курносова Н. С.** Развитие системы информационного обеспечения управления аграрным производством : дис. ... канд. экон. наук : 08.00.05 / Курносова Наталия Сергеевна. Воронеж, 2018. 167 с.
8. **Кушнарева М. Н.** Методические особенности определения эффективности внедрения информационных технологий на предприятии // Образование и наука без границ: фундаментальные и прикладные исследования. 2016. № 4. С. 45−49.
9. Обоснование требований к квалификации специалистов по управлению в отраслях и на предприятиях агропромышленного комплекса [Электронный ресурс]. URL: https://apknet.ru/obosnovanie-trebovanij.
10. О внесении изменений в состав сводной рабочей группы по совершенствованию положений Договора о Евразийском экономическом союзе от 29 мая 2014 года − ИПС «Әділет» [Электронный ресурс]. URL: https://online.zakon.kz/Document/?doc_id=37419732.

11. О продовольственной безопасности в 2013 году и прогнозе ее обеспечения в 2014 году [Электронный ресурс]. URL: http://government.ru/orders/12150.

12. **Петров К. А., Григорьев Н. С.** Сокращение производственных затрат и повышение рентабельности производства зерна на основе применения технологий точного земледелия (на примере Саратовской области) // Аграрный научный журнал. 2017. № 9. С. 93−96.

13. Постановление Правительства РФ от 9 октября 2021 г. № 1722 «О Федеральной государственной информационной системе прослеживаемости зерна и продуктов переработки зерна». https://www.garant.ru/products/ipo/prime/doc/402823063/.

14. Постановление Правительства Свердловской области от 28 июня 2019 г. № 386-ПП «Об утверждении Стратегии развития агропромышленного комплекса Свердловской области на период до 2035 года» (с изменениями и дополнениями) [Электронный ресурс]. URL: https://docs.cntd.ru/document/561427328.

15. Постановление Правительства РФ от 9 октября 2021 г. № 1722 «О Федеральной государственной информационной системе прослеживаемости зерна и продуктов переработки зерна» [Электронный ресурс]. URL: https://www.garant.ru/products/ipo/prime/doc/402823063/.

16. Приказ Министерства сельского хозяйства РФ от 17 ноября 2015 г. № 561 «Об утверждении плана информатизации Минсельхоза России на 2015 финансовый год и плановый период 2016 и 2017 годов» [Электронный ресурс]. URL: https://www.consultant.ru/document /cons_doc_LAW_225297/0e27c245d7e1754a42fbf9c74b6e4757f7d7642f.

17. **Худякова Е. В., Кушнарёва М. Н., Горбачев М. И.** Основные проблемы цифровой трансформации сельского хозяйства и пути их решения // Известия международной академии аграрного образования. 2022. № 62. С. 156−160.

18. **Худякова, Е. В., Горбачев М. И., Кушнарева М. Н.** Кадровой потенциал АПК в условиях цифровой трансформации // Новые информационные технологии в образовании: Сборник научных трудов 20-й международной научно-практической конференции, Москва, 04–05 февраля 2020 года / Под общей редакцией Д. В. Чистова. М. : ООО «1С-Паблишинг», 2020. С. 486−488. EDN ZCLDJF.

19. **Худякова Е. В., Кушнарёва М. Н., Горбачев М. И.** Эффективность внедрения цифровых технологий в соответствии с концепцией "сельское хозяйство 4.0" // Международный научный журнал. 2020. № 1. С. 80−88.

Учебное издание

Худякова Елена Викторовна
Степанцевич Марина Николаевна
Качалин Михаил Александрович
Горбачев Михаил Иванович

ЦИФРОВЫЕ ПЛАТФОРМЫ В АПК

Учебно-методическое пособие

Издается в авторской редакции

Оригинал-макет *Светлана Минченко*

Дизайн обложки *Роман Бурак*

Подписано в печать 14.03.2023. Формат 60x90/16
Усл.-печ. л. 5,87. Тираж 100 экз. Заказ № 6

ООО «Мегаполис»
www.m-megapolis.ru
Тел. 8 (495) 643-28-71
E-mail: mmegapolis-zakaz@yandex.ru
127550, Москва, ул. Прянишникова, д. 23 А

Отпечатано в ПАО «Т8 Издательские Технологии»
Тел.: +7 (499) 322-38-31
109316, Москва, Волгоградский проспект, д. 42, корп. 5