УДК 303.732.4

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ОРГАНИЗАЦИИ И ПРОВЕДЕНИЯ ФУТБОЛЬНЫХ МЕРОПРИЯТИЙ ПО «НИДЕРЛАНДСКОЙ СИСТЕМЕ»

Барков Евгений Анатольевич[1], Лукьянов Константин Валерьевич[2]

[1]Системный аналитик;
X5 Group («Vprok.ru Перекресток»);
Россия, 109029, г.Москва, ул.Средняя Калитниковская, д.28, стр.4;
e-mail: barkoffevgen@gmail.com.

[2]Старший научный сотрудник;
Объединенный институт ядерных исследований;
Россия, 141980, Московская обл., г.Дубна, ул. Жолио-Кюри, д. 6;
Кандидат физико-математических наук, доцент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г.Дубна, ул. Университетская, д. 19;
e-mail: luky@jinr.ru.

В работе представлена концепция проведения детских футбольных турниров и соревновательных фестивалей в формате «нидерландской системы». Для реализации данной концепции была спроектирована информационная система, позволяющая автоматизировать процесс взаимодействия лиц, принимающих участие в организации спортивного мероприятия в формате рассматриваемой концепции.

Ключевые слова: нидерландская система, проектирование, сбор требований, детско-юношеский футбол.

Для цитирования:

Барков Е. А., Лукьянов К. В. Проектирование информационной системы для организации и проведения футбольных мероприятий по «Нидерландской системе» // Системный анализ в науке и образовании: сетевое научное издание. 2022. №3. С.180-190. URL:http://sanse.ru/download/478.

DESIGNING AN INFORMATION SYSTEM FOR ORGANIZING AND CONDUCTING FOOTBALL EVENTS ACCORDING TO THE "DUTCH SYSTEM"

Barkov Evgeny A.[1], Lukyanov Konstantin V.[2]

[1]Systems analyst;
X5 Group («Vprok.ru Perekrestok»);
28 Sredniaia Kalitnikovskaia Str., Moscow, 109029, Russia;
e-mail: barkoffevgen@gmail.com.

[2]Senior Researcher;
Joint Institute for Nuclear Research;
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;
PhD in Physical and Mathematical Sciences, associate professor;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: luky@jinr.ru.

The article presents the concept of holding children's football tournaments and competitive festivals in the format of the "Dutch system". To implement this concept, an information system was designed to automate the process of interaction of persons participating in the organization of a sports event in the format of the concept under consideration.

Keywords: Dutch system, designing, collecting requirements, children football.

For citation:
Barkov E. A., Lukyanov K. V. Designing an information system for organizing and conducting football events according to the "Dutch system". *System analysis in science and education*, 2022;(3):180-190(in Russ). Available from: http: //sanse.ru/download/478.

Введение

Тема детско-юношеского спорта может рассматриваться под разным углом. Занятия спортом развивают не только физические навыки, но и социальные, что является важной составляющей при становлении человека, как личности. Существует много разных механик, с помощью которых удается привлекать подрастающее поколение к занятиям разными видами спорта. В случае с футболом большое влияние на детей оказывают эмоции, которые проявляются в процессе игры в мяч.

Проведение детских футбольных турниров – тяжелая задача с точки зрения временных и физических затрат. Тем не менее, например, в Нидерландах, в стране, которая является одной из лучших в мире с точки зрения организации детско-юношеского футбола, крупные организации каждую неделю проводят футбольные фестивали для мальчиков и девочек. Используются разные форматы игр: от классических, до специальных [1]. Один из специальных форматов, называемый «нидерландской системой», предусматривает регулярную смену игроком команды после каждого матча [2]. Это позволяет ребенку быть более вовлеченным в игровой процесс, избегая при этом сложностей, связанных с классическим форматом проведения игр. Разберем эти особенности подробнее.

Концепция «нидерландской системы» при проведении футбольных фестивалей

Основные правила:

1. Игры проходят в малых формах (в составе находится от 2 до 4 человек).
2. Короткие таймы (5–7 минут);
3. Каждый тур команды перемешиваются, игрок начинает следующий матч с новыми партнерами по команде.
4. По ходу турнира подсчитывается индивидуальный рейтинг футболиста (на основе побед, поражений, ничьих и забитых голов).
5. Победителем турнира является не команда, а конкретный человек, занявший первое место в рейтинге.

Отличия «нидерландской системы» от классических турниров:

1. Игры проходят в малых форматах, что позволяет ребенку за игровой промежуток времени больше коснуться мяча, нежели если бы юный спортсмен играл в команде, где больше участников в составе.
2. Ребенку не нужно находить команду, чтобы поиграть в футбол. Достаточно приехать на мероприятие, пройти процесс регистрации и начать играть.
3. Двойная ответственность. Количество полученных очков по итогам матча зависит не только от того, сколько индивидуальных действий совершит спортсмен, но и от результата всей команды.
4. Ребенок, который, как кажется, не проявляет себя ярко в игре, имеет шансы занять высокие места в итоговом рейтинге.
5. Каждый юный спортсмен гарантировано получает одинаковое количество игрового времени.

Однако, такие особенности «нидерландской системы» несут в себе и ряд организационных проблем, с которыми сталкиваются устроители турниров:

1. Заполнение результатов. Так как судьи находятся друг от друга на большом расстоянии, результаты матчей собираются только в конце тура. Их заполнение на бумажном носителе и последующая синхронизация требуют время, которое в сумме потенциально может использоваться для проведения еще нескольких игр.
2. Вероятность ошибок. Человек может ошибиться во время подсчета очков у каждого участника из-за большого количества разных числовых данных.
3. Формирование итоговой турнирной таблицы. Участники мероприятия узнают об итогах своих выступлений только в конце турнира, так как организаторы не имеют временных возможностей считать рейтинг после каждого матча, не говоря о подсчете рейтинга в режиме реального времени.

Решить указанные выше проблемы можно с помощью информационной системы, которая может быть реализована, например, в виде мобильного приложения. В данной статье рассмотрены основные этапы проектирования информационной системы для автоматизации процесса взаимодействия лиц, организующих футбольный турнир или соревновательный фестиваль по «нидерландской системе».

Анализ сценариев проведения фестивалей по «нидерландской системе»

На иллюстрации «*AS IS*» изображен сценарий, где, если рассматривать с организаторской точки зрения, имеется:

- 4 площадки;
- 4 судьи;
- организаторский штаб, в котором происходит ручной подсчет рейтинга на основе результатов матча.

В случае «*AS IS*» отсутствует какая-либо автоматизация. Судьи и организаторы взаимодействуют только в конце игровых сессий. Перерывы между матчами длинные, так как:

- затрачивается время на проход судей от площадки до организаторского штаба;
- требуется время на одновременную обработку результатов игровой сессии со всех площадок;
- требуется время на подсчет очков для турнирной таблицы и изменение расположения игроков в ней.

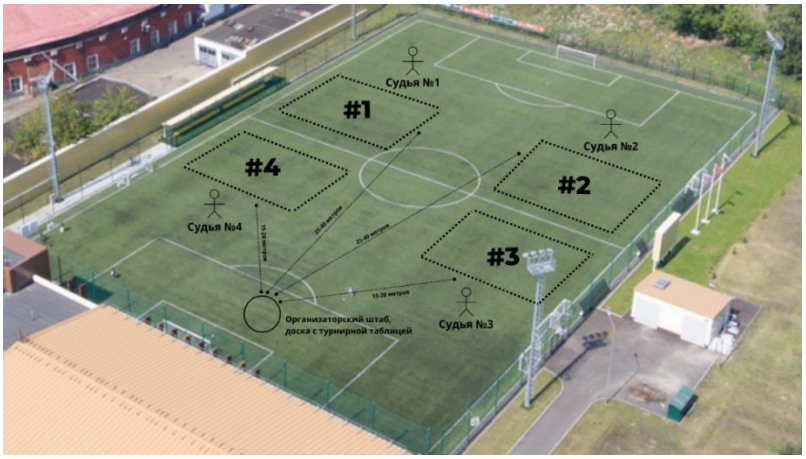

Рис. 1. Пример процесса организации турниров по «нидерландской системе» «AS IS»

Сценарий использования проектируемой информационной системы «*TO BE*» подразумевает наличие мобильных устройств у лиц, организующих футбольное мероприятие. К ним относятся судьи и организаторы. На рис. 2 изображен примерный процесс одновременной передачи информации судьями о площадках в информационную систему, которая затем обрабатывает результаты и отправляет организатору турнирную таблицу в режиме реального времени.

Рис. 2. Пример процесса организации турниров по «нидерландской системе» «TO BE»

Описание процессов на «верхнем уровне»

На основании сценария использования ИС «*TO BE*» сформулируем процессы верхнего уровня при помощи языка *BPMN* [3], являющегося промежуточным звеном между формализацией, визуализацией и воплощением бизнес-процесса. Результат представлен на рис. 3.

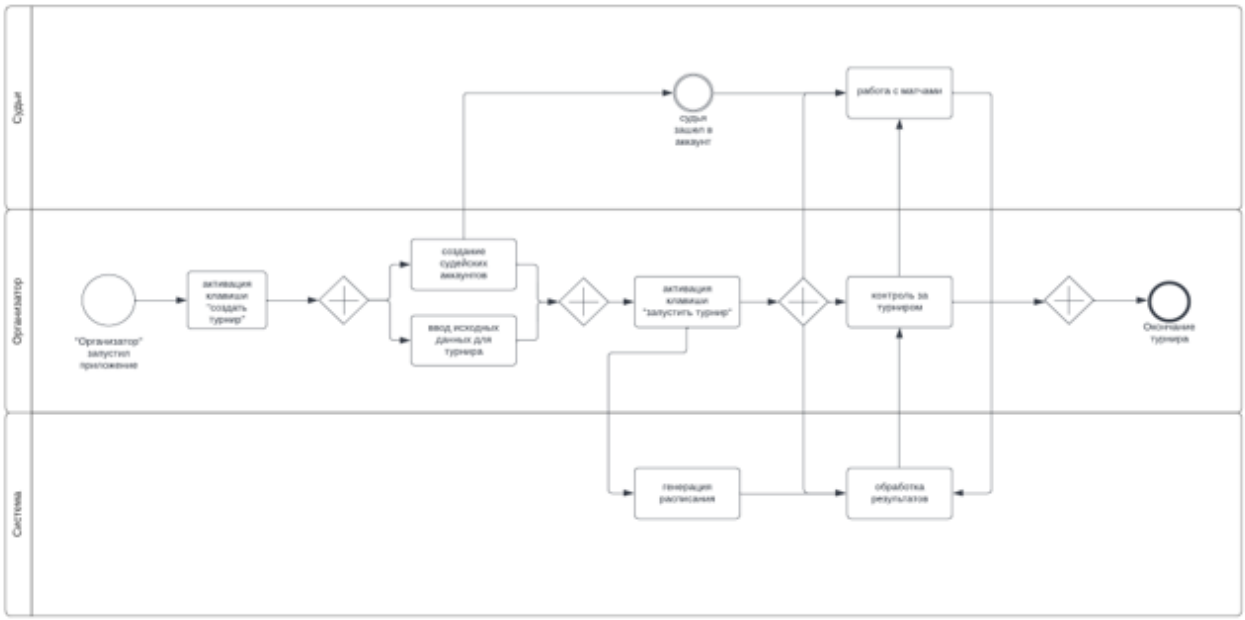

Рис.3. BPMN-диаграмма проектируемой информационной системы

Разделение системы на отдельные компоненты и структура формирования требований

Декомпозиция основных процессов системы

Для эффективного проектирования информационной систем разделим основные процессы на подпроцессы, то есть проведем декомпозицию [4]. На основе *BPMN*-диаграммы (рис. 3) выделим несколько ключевых функциональных блоков:

1. «Организатор» вводит исходные данные для генерации расписания турнира (запуска турнира), создает судейские аккаунты;
2. «Организатор» контролирует ход турнира;

3. «Судья» взаимодействует с «панелью управления матчем».

Для каждого из перечисленных блоков сформулируем требования в соответствии с нижеследующей структурой.

Структура формирования требований к подпроцессам информационной системы

Требования, которые формируются с целью проектирования информационной системы, разделяются по характеру: функциональные и нефункциональные. Функциональные требования – это требования к поведению системы. Нефункциональные требования – это требования к характеристикам системы, качеству, ограничению.

Для проектирования подпроцессов были сформированы функциональные требования, в частности: пользовательские и системные требования. В статье будет частично разобран процесс сбора требований к функциональному блоку «Организатор вводит исходные данные для генерации расписания турнира (запуска турнира), создает судейские аккаунты».

Описание функционального блока: «Организатор вводит исходные данные для генерации расписания турнира (запуска турнира), создает судейские аккаунты»

Блок-схема

Последовательность действий организатора можно описать следующей блок-схемой.

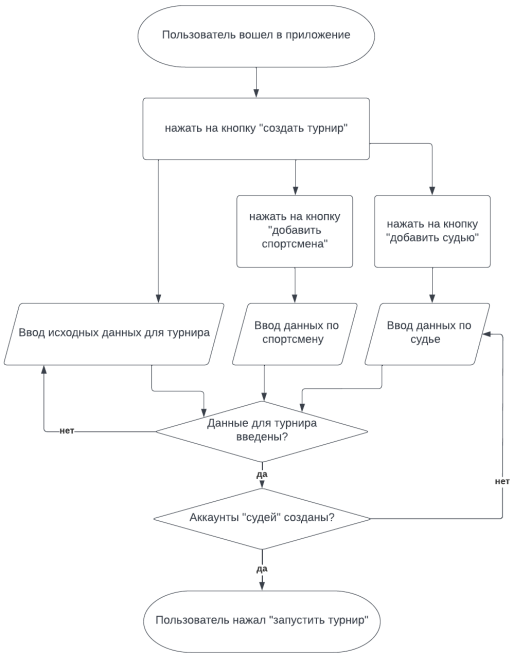

Рис. 5. Блок-схема, описывающая функциональный блок «Организатор вводит исходные данные для генерации расписания турнира (запуска турнира), создает судейские аккаунты»

Пользовательские требования. User Story

Пользовательские требования – это требования, определяющие набор пользовательских задач, которые должна решать информационная система. Самым простым вариантом описания пользовательских требований является *User Story* [5]

Для рассматриваемой информационной системы будем придерживаться следующей структуры:

1. Объект, вокруг которого описывается *User Story*, например: «Я, как пользователь».

2. Описание возможности, которую хочет получить объект, например: «Хочу иметь возможность делать что-то».
3. Описание ценности, которую хочет получить после представленной возможности, например: «Чтобы получить что-то».

Пример: «Я, как приложение, хочу иметь возможность отправлять *push*-уведомления, чтобы пользователь быстро получал информацию». Описание «пользовательских историй» для компонентов системы было реализовано в виде специальных таблиц (Таблица 1):

Табл. 1. Пример описания User Story для рассматриваемого функционального блока в виде специальной таблицы

Требование	***User Story***	**Критерии приемки**
Возможность добавления исходных данных для турнира	Я, как пользователь, хочу иметь возможность добавлять исходные данные для турнира, чтобы система сгенерировала расписание.	Я могу добавить в специальные формы необходимые данные. На экране есть формы: - ввести кол-во участников - кол-во игровых площадок - длительность турнира - длительность тайма - кол-во человек в команде
Возможность создания судейских аккаунтов	Я, как пользователь, хочу иметь возможность создавать аккаунты для судей, чтобы судья, который привязан к площадке, получил доступ к системе	Судья, привязанный к площадке, получил доступ в аккаунт, а также получил стартовый экран для судейского аккаунта
…	…	…

Системные требования. Use Case

Системные требования – более детализированное описание пользовательских требований, в контексте углубления в технические нюансы. Одним из способов описания взаимодействия пользователя с ИС является *Use Case* [6].

Описание «сценариев использования» для компонентов системы было реализовано в виде специальных таблиц (Табл. 2):

Табл. 2. Пример описания одной из функций в рассматриваемом функциональном блоке в виде Use Case

Действующие лица	1. Пользователь – «Организатор» 2. Приложение – клиентское приложение 3. Сервер
Цель	Добавить данные для создания турнира и сгенерировать судейские аккаунты
Триггер	Нажатие на кнопку «создать турнир»
Поток событий:	
1. Приложение – выводит экран с формами ввода данных и кнопками «добавить судью», «добавить участника» 2. Пользователь – вводит данные в форму ввода количества участников турнира 3. Пользователь – вводит данные в форму ввода количества игровых площадок 4. Пользователь – вводит данные в форму ввода «длительность турнира» 5. Пользователь – вводит данные в форму ввода «длительность тайма»	

6. Пользователь – вводит данные в форму ввода «количество человек в команде 7. Пользователь – нажимает на кнопку «добавить судью» 8. Приложение – выводит экран с формами ввода данных и кнопкой «добавить судью» 9. Пользователь – создает необходимое количество судейских аккаунтов 10. Пользователь – нажимает на кнопку «назад» 11. Приложение – выводит экран с формами ввода данных и кнопками «добавить судью», «добавить участника» 12. Пользователь – нажимает на кнопку «запустить турнир» 13. Приложение – проверяет, введены ли данные 14. Приложение – отправляет данные на сервер для дальнейшей генерации расписания
Результат – Пользователь ввел исходные данные для создания турнира
Альтернативный сценарий: пользователь внес не все данные
7. Пользователь – нажимает на кнопку «добавить судью» 8. Приложение – проверяет, введены ли данные 9. Приложение – выводит окно с ошибкой

Архитектурная концепция

Для описания взаимодействия действующих лиц в процессах задействуем *UML Sequence-диаграмму* [7]. Пример *Sequence*-диаграммы на рисунке ниже (рис. 6):

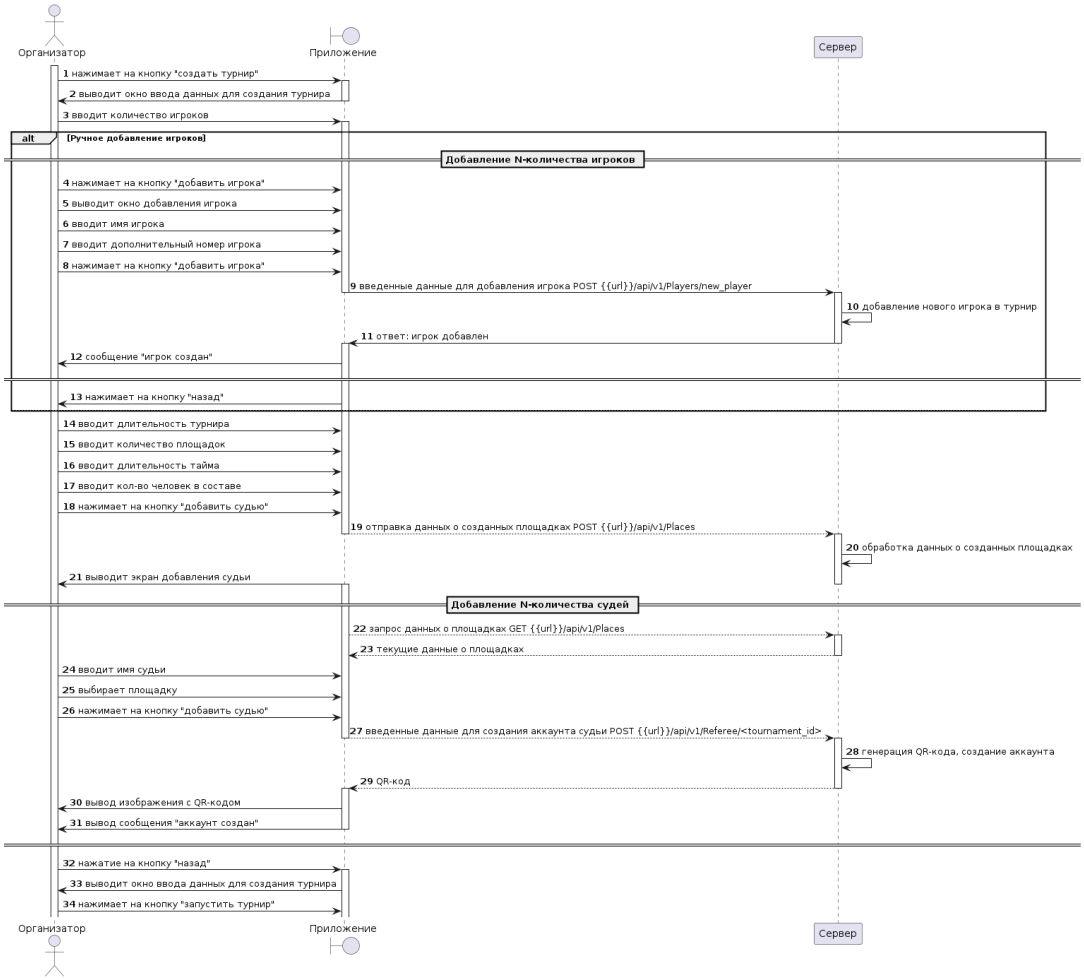

Рис. 6. UML Sequence-диаграмма, на которой изображен функциональный блок: «Организатор» вводит исходные данные для генерации расписания турнира (запуска турнира), создает судейские аккаунты

Табл. 3. Пояснение к шагам, упомянутым в UML-Sequence диаграмме

Описание процесса	**Пояснение**
№1 - «Организатор» нажимает на кнопку «создать турнир»	Пользователь («Организатор») находится на стартовой странице и нажимает на кнопку «создать турнир»
№2 – Приложение выводит окно ввода данных для создания турнира	Приложение после нажатия пользователем кнопки «создать турнир» загружает окно ввода данных для создания турнира
№3 – «Организатор» вводит количество игроков	Пользователь («Организатор») нажимает на форму ввода количества игроков (участников турнира), вводит числовое значение
№4 – «Организатор» нажимает на кнопку «добавить игрока»	Пользователь («Организатор») нажимает на кнопку «добавить игрока», чтобы перейти на экран ручного добавления игроков
№5 – Приложение выводит экран добавления игрока	Приложение отображает экран ручного добавления игрока
№6 – «Организатор» вводит имя игрока	«Организатор» нажимает на форму ввода имени игрока Форма активируется и «Организатор» вводит имя игрока
№7 - «Организатор» вводит дополнительный номер игрока	«Организатор» нажимает на форму ввода дополнительного номера игрока Форма активируется и «Организатор» вводит дополнительный номер игрока
№8 - «Организатор» нажимает на кнопку добавить игрока	«Организатор» активирует процесс обработки полей, куда были введены данные об игроке
№9 – Введенные данные для добавления игрока, *POST {{url}}/api/vi/Players/new_player*	Взаимодействие клиент-сервер. «Клиент» (Приложение) отправляет *POST* запрос, где «Сервер» получит данные для добавления игрока в систему.
№10 – Добавление нового игрока в турнир	Выгрузка «Сервером» данных из полученного *JSON*-документа, маппинг полученных полей, добавление игрока в систему и сохранение данных в базу данных.
…	…

Взаимодействие клиент-сервер

Проектируемое приложение предполагает использование клиент-серверной архитектуры. В контексте рассматриваемой нами информационной системы «сервер» — это некий объект в сети, который способен принять *HTTP*-запросы, обработать их и вернуть ответ. «Клиент» — это некий объект, способный отправлять *HTTP*-запросы [8,12].

Взаимодействие «клиент-сервер» будет осуществляться при помощи *API* - механизма, дающего возможность взаимодействовать двум отдельным программным сущностям между собой [9,10,11].

В ходе проектирования информационной системы были разработаны методы *API*, через которые компоненты приложения взаимодействуют с сервером.

Пример описания методов API

Метод: отправка данных для создания турнира

- *URL*-адрес - {{*url*}}/*api*/*v*1/*Tournament_Data*/<*tournament_id*>;
- Метод операции – *POST*;
- Назначение метода - метод получает данные для создания турнира (генерации расписания).

Бизнес-логика метода:

Метод вызывается, когда пользователь нажимает на кнопку «запустить турнир». Клиентское приложение отправляет введенные пользователем данные по количеству участников, длительности турнира, количеству площадок, длительности тайма и количеству участников в команде.

Параметры запроса (таблица 4):

\

Табл. 4. Параметры запроса метода «Отправка данных для создания турнира»

Параметр	**Req**	**Тип данных**	**Описание**	**Источник данных**
<tournament_id>	*не может быть Null*	*integer*	Поле, которое хранит в себе уникальный номер создаваемого турнира	*Tournament_Data.tournament_id*
sum_of_players	*не может быть Null*	*integer*	Поле, которое хранит в себе количество игроков, которых система должна добавить автоматически в создаваемый турнир	*Tournament_Data.sum_of_players*
sum_of_places	*не может быть Null*	*integer*	Поле, которое хранит в себе количество игровых площадок	*Tournament_Data.sum_of_places*
timing	*не может быть Null*	*integer*	Поле, которое хранит в себе длительность турнира	*Tournament_Data.timing*
set_timing	*не может быть Null*	*integer*	Поле, которое хранит в себе длительность тайма	*Tournament_Data.set_timing*
sum_of_players_in_team	*не может быть Null*	*integer*	Поле, которое хранит в себе количество игроков в каждой команде	*Tournament_Data. sum_of_players_in_team*

Примеры запроса и ответа (таблица 15):

Табл. 5. Пример запроса и ответа к методу «Отправка данных для создания турнира»

Запрос	Ответ

{
"tournament_id": 12,
"sum_of_players": 30,
"sum_of_places": 5,
"timing": 60,
"set_timing": 5,
"sum_of_players_in_team": 3
}
	200 OK

Заключение

В настоящей статье представлены примеры сбора требований к информационной системе для организации и проведения футбольных мероприятий (турниров) по «нидерландской системе», в частности описан один из функциональных блоков программного продукта, а также функциональные и нефункциональные требования. Для остальных функциональных блоков подобные, и даже более подробные выкладки даны в выпускной квалификационной работе «Проектирование информационной системы для организации и проведения футбольных мероприятий по «нидерландской системе» [13], также размещенной в свободном доступе в сети интернет.

Существует определенное понимание, каким образом разработанный проект потенциально может развиваться и масштабироваться. На рис. 8 изображен перспективный жизненный цикл информационной системы для проведения и организации турниров по «нидерландской системе»:

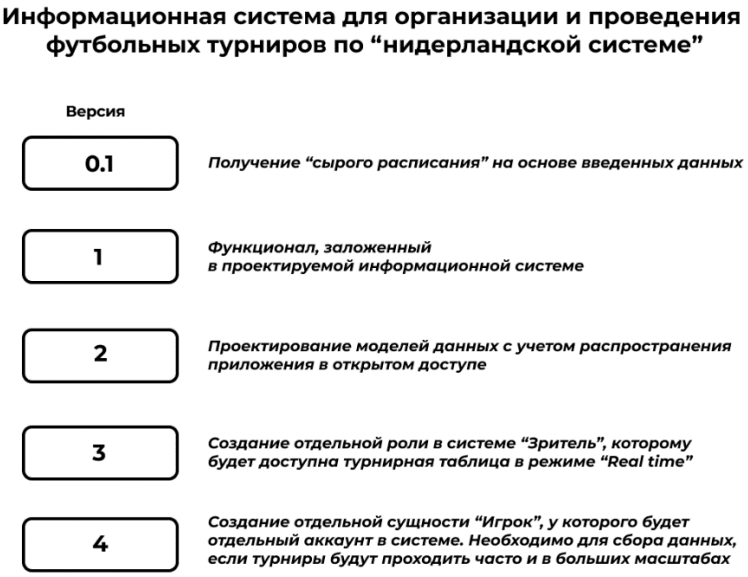

Рис. 8. Перспективные функции информационной системы для организации и проведения турниров по «нидерландской системе»

Список источников

4. KNVB Oranje Fandag // Players United – One Game. All United [:веб-сайт]. PLAYERS UNITED, 2022. URL: https://playersunited.com/knvb-oranje-fandag/ (дата обращения: 20.05.2022).
5. Internal game format 2 versus 2 under 6s (introduction to football) as of the 2017/2018 season : recommendation // Dutch Youth Football. KNVB, 2017. URL: https://knvb.h5mag.com/dutch_youth_football/u6-recommendation (дата обращения: 03.04.2022).

6. Нотация BPMN // Business Studio Wiki. Business Studio: проектирование организации [:веб-сайт]. Группа компаний «Современные технологии управления», 2004–2022. URL: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/bpmn_notation (дата обращения: 03.05.2022);

7. Agile in IT: 8 методов декомпозиции задач // doITsmartly - профессиональное развитие в IT [:веб-сайт]. doITsmartly, 2022. URL: https://doitsmartly.ru/all-articles/management/99-agile/117-decomposition-techniques.html (дата обращения: 20.04.2022).

8. Кустов Д., Баранов Р. User Story – инструкция по применению // ScrumTrek [: блог]. ScrumTrek, 2008–2022. Дата публикации 11 августа 2020. URL: https://scrumtrek.ru/blog/product-management/3364/user-story-instruktsiya-po-primeneniyu/ (дата обращения: 12.04.2022).

9. Что такое use case? Теория и примеры // testengineer.ru – тестирование, бизнес-анализ, проджект –менеджмент [:веб-сайт]. testengineer.ru, 2020–2022. Дата публикации 23 ноября 2021. URL: https://testengineer.ru/chto-takoe-use-case/ (дата обращения: 15.04.2022).

10. Иванов Д. Ю., Новиков Ф. А. Основы моделирования на UML: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2010. – 249с.

11. Лекция 4. Протокол HTTP // www.4stud.info. Учебно-методические материалы для студентов кафедры АСОИУ [:веб-сайт]. www.4stud.info, 2007 – 2022. URL: https://www.4stud.info/web-programming/protocol-http.html (дата обращения: 20.04.2022).

12. Введение в web APIs / Сообщество MDN Web Docs // MDN Web Docs. Resources for Developers, By Developers [:веб-сайт]. Individual mozilla.org contributors, 1998–2022. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата обращения: 18.04.2022).

13. Введение в REST API – RESTful веб-сервисы // Хабр. Habr, 2006–2022. Дата публикации 10 января 2020. URL: https://habr.com/ru/post/483202/ (дата обращения: 21.05.2022).

14. Data Mapping: лучшие техники и инструменты // MAD блог: О данных, маркетинге и бытие. MAD Data Agency, 2020. Дата публикации 12 ноября 2021. URL: https://maddata.agency/blog/data-mapping-luchshie-tekhniki-i-instrumenty свободный (дата обращения: 26.05.2022).

15. Лекция 6, ч. 1.Архитектура клиент-сервер // Курс лекций "Тестирование програмного обеспечения". URL: https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-6-ch1-arhitektura-klient-server.html (дата обращения: 19.05.2022).

16. Барков Е. А. Проектирование информационной системы для организации и проведения футбольных мероприятий по "нидерландской системе" : Бакалаврская работа / Е. А. Барков, руководитель работы К. В. Лукьянов; рецензент П. П. Сычев. Дубна: государственный университет "Дубна", 2022.