Задача 6. Да се направи декомпозиция на модулите от архитектурата на софтуерна система за организиране на онлайн залагания на спортни събития, според дадените по-долу изисквания. Обосновете защо така проектираната архитектура удовлетворява изискванията.

1. Клиенти на системата са различни организатори на залагания, които представляват юридически лица.
2. Залагания може да се правят само от регистрирани потребители.
3. Нерегистрирани потребители могат да преглеждат списъкът с налични събития за залагане и съответните предложения за коефициенти на залозите към тях.
4. Коефициентите на залозите могат да се променят за секунди, като се изчисляват по математическа зависимост, която се определя количествено както от честотата на направените залози до момента, така и от прогнозни резултати.
5. Прогнозните резултати се взимат от външна услуга, която се достъпва по специфичен протокол.
6. Залаганията и печалбите не са в реални пари, в т.нар. кредити, които всеки потребител разполага в акаунта си. Кредитите може да се зареждат по различни начини, които се определят индивидуално за всеки клиент.
7. Потребителите могат да правят неограничен бтой залагания, но само в рамките на текущо наличните им кредити.
8. Системата е непрекъснато свързана с онлайн регистър на данъчната служба в съответната държава, където е регистриран клиентът.
9. Системата трябва да е достъпна 24/7, тъй като потребителите могат да се включват от различни часови зони.
10. Системата трябва да е защитена от външна намеса.

Решение:

Описаната по-горе система и изисквания показват, че тя ще се използва както от юридически лица, като например bet365, eurosport и т.н., така и от физически такива (крайни потребители). За да може да използват главните функционалности на този софтуер е необходимо те да бъдат регистрирани. Тоест, това очертава рамките на първия модул "Registration". За да може да се направи тази регистрация, възниква въпроса дали даденото юридическо или физическо лице изобщо съществува. Това би могло да се валидира чрез външна система на НАП или ЕСГРАОН съответно. Регистрацията ще отключва иначе забранени функционалности на системата, докато други ще са достъпни дори и без регистрация. Очевидно видовете регистрация ще са минимум два. Една от основните функционалности ще е преглеждането на опциите за залагания, която ще е достъпна и без регистрация и влизане в профила. Тези опции ще ги опаковаме в модула "Gambling List" (допълнително може да се върнем в този модул и да го декомпозираме рекурсивно надолу до разграничаване на секции). За изчисляването на коефициентите ще се грижи друг модул, който ще наречем "Gambling Algorithms". Той от своя страна според бизнес логиката ще се грижи да определя какви да бъдат коефициентите за залагане, така че юридическите ни потребители на системата да не са на загуба в никакъв сценарии, иначе ще се лишим от тяхното желание да използват системата ни. "Gambling List" ще си комуникира с още един модул "Bet", който е опаковал една друга основна функционалност на системата, валидна само за регистрирани физически лица, а именно самото залагане. Сега, възниква следния казус – кое ще се изпълни първо – актуализацията на коефициента за залога или приемането на залога от потребителя? За целта ще обособим друг модул "Gambling Scheduler", който ще играе роля на разпределител или някакъв диспетчер, тъй като за тези процеси явно ще има борба за ресурси. Но нека се абстрахираме от това и допуснем, че операционната система ще се грижи за този проблем. Но тогава пък възниква друг въпрос – отлагане на свързването, тоест за всяка операционна система ще трябва да има обособен crack, patch или някакъв друг допълнителен софтуер, който да приспособява нашата система (явно или неявно за потребителя) към съответната операционна система. Това ще добави твърде много модули и ще лиши системата ни от контрол до някаква степен, за това нека се върнем на първоначалната идея с допълнителния контролер/разпределител. Тъй като прогнозите ще се взимат от някаква външна услуга, то нашата система ще си комуникира с API-то на тази външна услуга и то ще си комуникира с "Gambling Algorithms", която комуникация ще е предоставена със

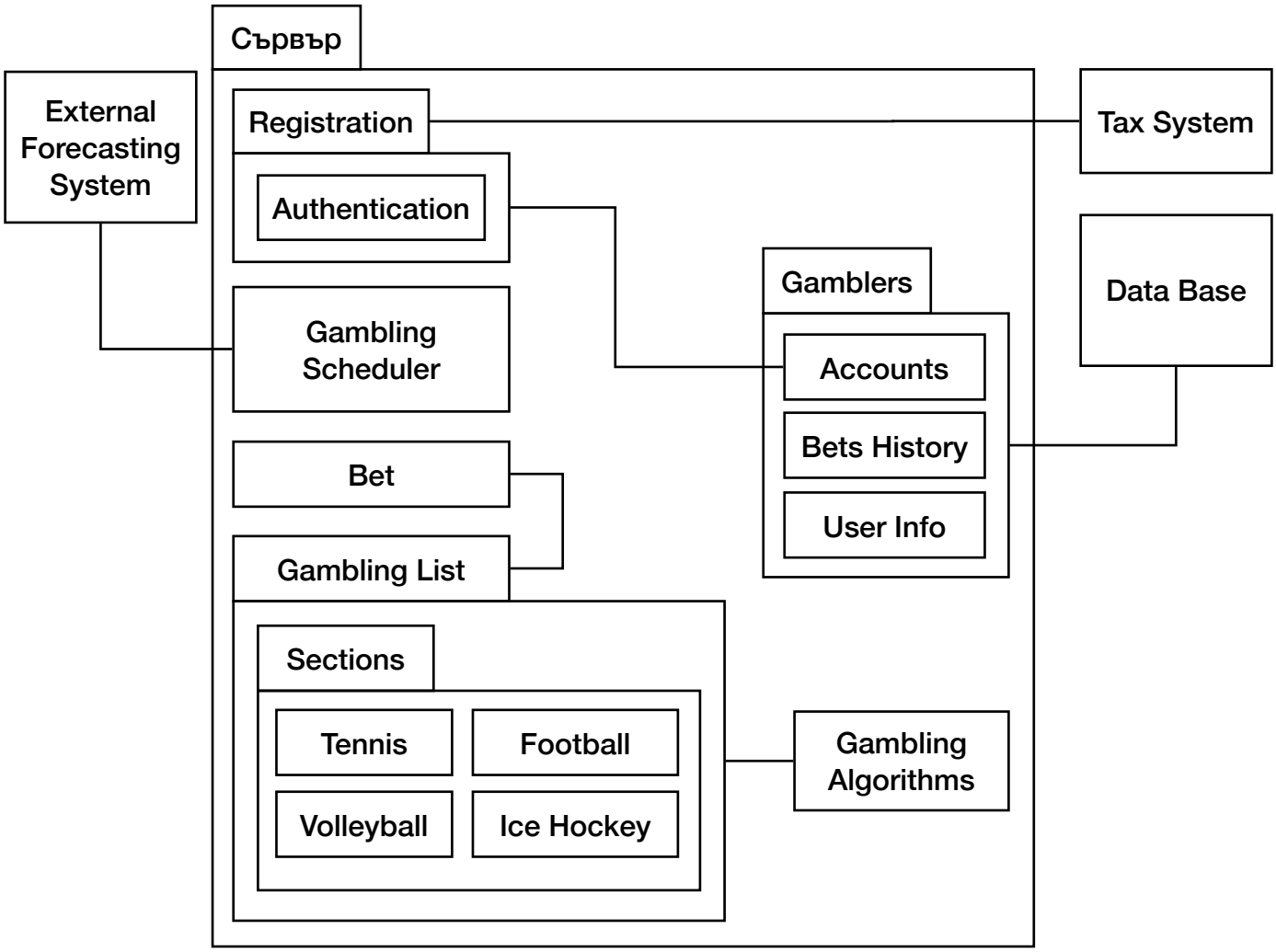
специфичен протокол. Сега, за да отговорим на следващото изискване – ще направим модул "Gamblers", в който ще има подмодули "User Info", "Account" и "Bets History". В него ще се зарежда информация за наличната сума на потребителя, както и на какво най-много обича да залага, колко и какъв тип залози прави, тоест нещо като профил и история за него, но там ще се допитваме и когато той желае да направи някакъв залог и дали има необходимите средства.
Тъй като системата трябва да е достъпна 24/7, то ще се наложи да реплицираме сървърите и различни сървъри да отговарят на различни заявки. Например сървър за тенис залагания, сървър за футболни залагания и т.н.. Може дори разделението да е по съвсем различен критетии – спрямо големината на залога, например. За да гарантираме, че няма да има загуба на данни при отказ, тоест да скрием отказа, ще поддържаме активен излишък като направим дубликати на базата данни и я поддържаме активна при всяко обновление (ще използваме и журнали за улеснение). Това драстично ще намали производителността, но тук може и да се върнем да помислим още.
Нека разгледаме следващото изискване: За защита от външна намеса най-вероятно се има предвид DoS или DDoS атаките. Нека за улеснение разгледаме само първата.

- За целта ще въведем автентикация на потребителите. Например, ако само регистриран потребител може да използва дадени услуги на системата, то при регистрация ще трябва да разпознава в кое от квадратчетата на подадена снима има... част от жираф, например след което да въвежда паролата си и други данни, които е предоставил при първоначалната си регистрация.
- Въвеждането на ограничение на експлоатацията на местата където са уязвими от атака. Например ако сървърът може да обработва по 1000 заявки за някакво изчисление, то всяко следващо ще чака. Но това пак не разрешава проблема с DoS SYN Flood типа досовска атака, където нападателят може да започва процедура по залагане но да оставя заявката без отговор и така те остават незавършени и заемат мрежови ресурс. За това може да се въведе и времеви лимит за изпълнението на всяка заявка, след което тя да се прекратява автоматично.

Някой по-сериозни мерски, които може да предложим и комбинираме:

- Наблюдаване на трафика с цел установяване възможно най-рано на DoS атаката. За целта трябва да знаем какво означава ниско, средно и високо натоварване на системата, като отчитаме сезонност, маркетингови кампании и т.н.. Така ще минимизираме загубите.
- Ще ограничим достъпа с допълнителен слой защита чрез средствата за ограничаване на по-нататъшен достъп като DMZ или firewall. Може при необходимост дори да използваме двойни защитни стени (периметърна и вътрешна).
- Ако допуснем, че приложението генерира сериозен интерес и трафик от различни части на света, то ще съхраняваме данните на няколко сървъра по целия свят. Това го правят именно CDN мрежите. Те услужват данните към потребителите от сървър, който е най-близо до тях с цел по-бърза връзка. Това автоматично прави системата по-малко уязвима на такива атаки, защото има много други сървъри, които функционират.

Диаграма може да се направи по произволен начин, спрямо по-горе описаните модули и свръзки. Вариантите са десетки. Ето един от тях:



Коментар: Декомпозицията на модулите е от високо ниво. Системата от условието е твърде сложна, за да се навлиза в детайли. Навлезли сме в повече детайли в обосновката на архитектурата.
Задачата е давана на второ контролно по САРС на 28 май 2020 г. от доц. д-р Александър Димов (който най-вероятно е и нейн автор от ДИ). На задачата можеха да се получат максимум 10 точки. За 80% от описанието предоставено по-горе бях получил 13 точки (с 3 над максимума). След контролното доцента спомена, че много му е допаднала частта в която се размишлява и се сменя решението за Scheduler-а. Тоест, съпоставката между няколко стратегии или видове декомпозиция и аргументирането на избора е огромен фактор в оценяването.