

A – nie, Controller wykonuje operacje na obiektach Modelu

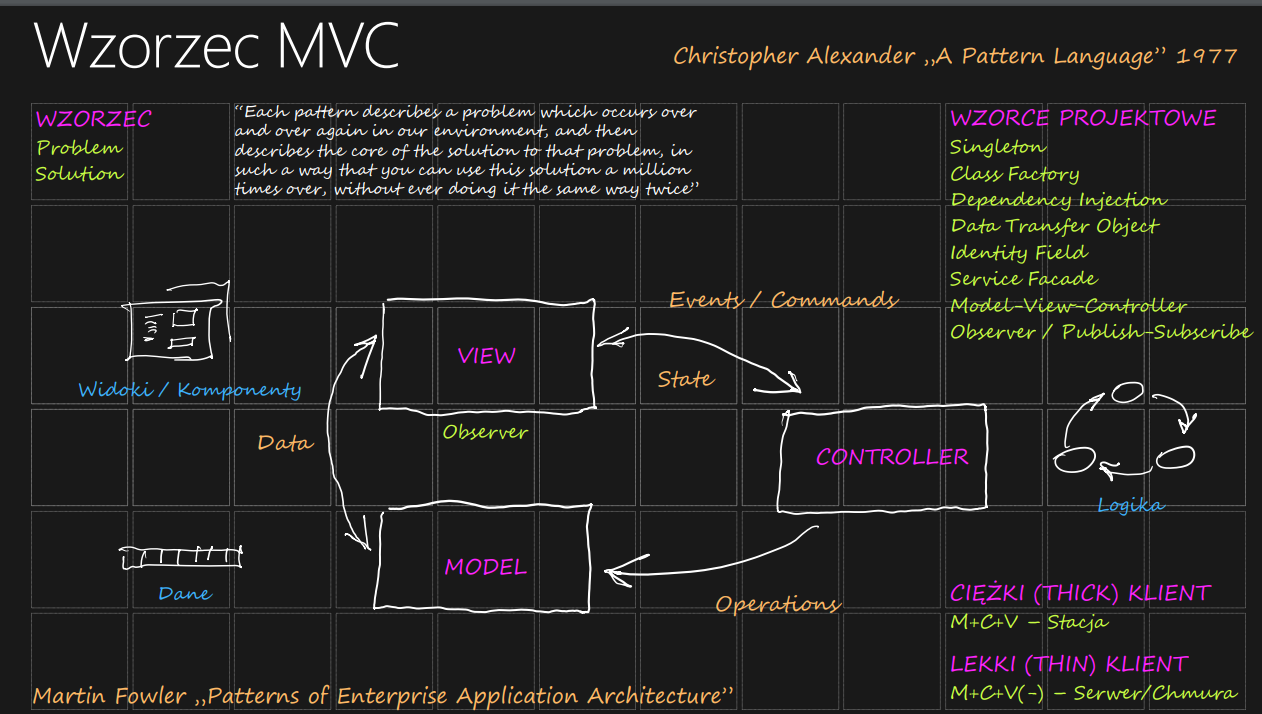
B - nie?

C – tak

D – cringe idk, lepiej nie

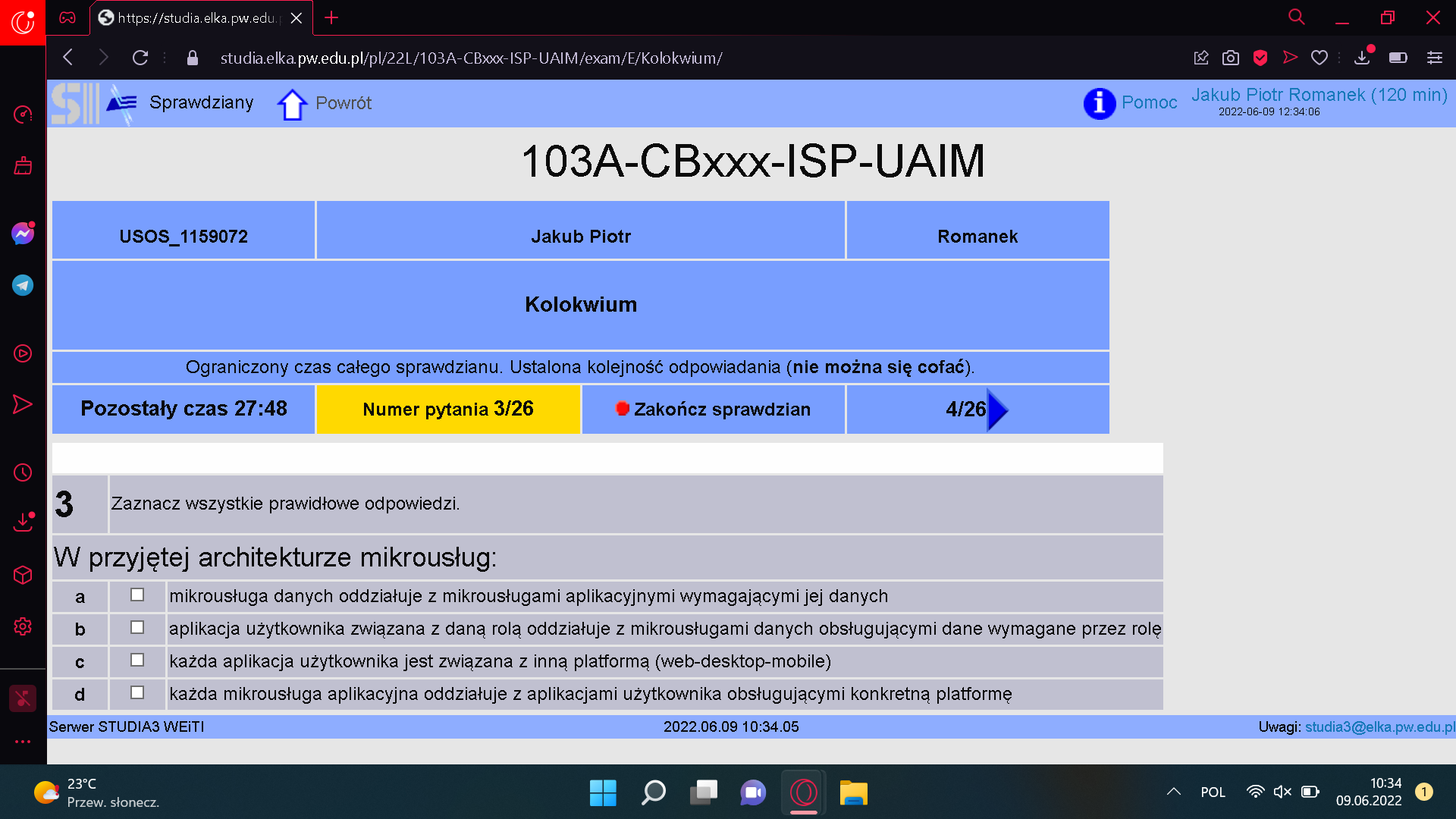
The Model is known as domain object or domain entity. //model == domena, obiekt domeny

//*bezpiecznie zaznaczyć C tylko*



Obraz zawierający tekst

Opis wygenerowany automatycznie



A ok

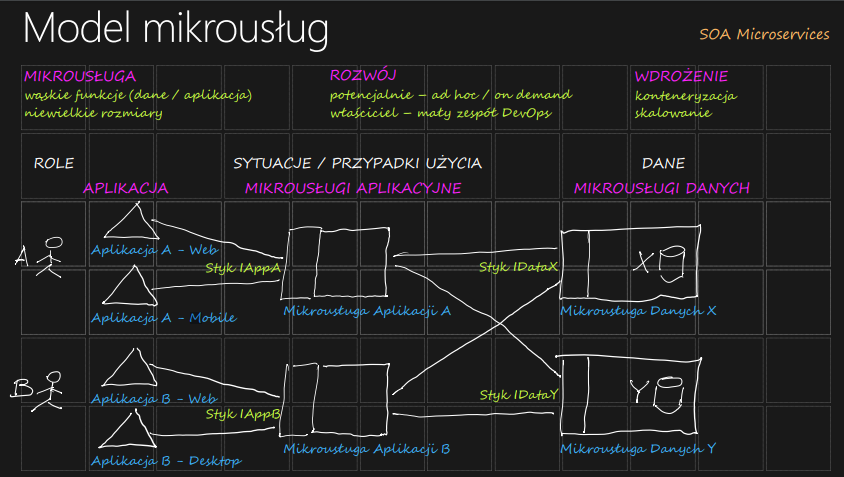
B nie, pomiędzy jeszcze mikrousługa aplikacyjna

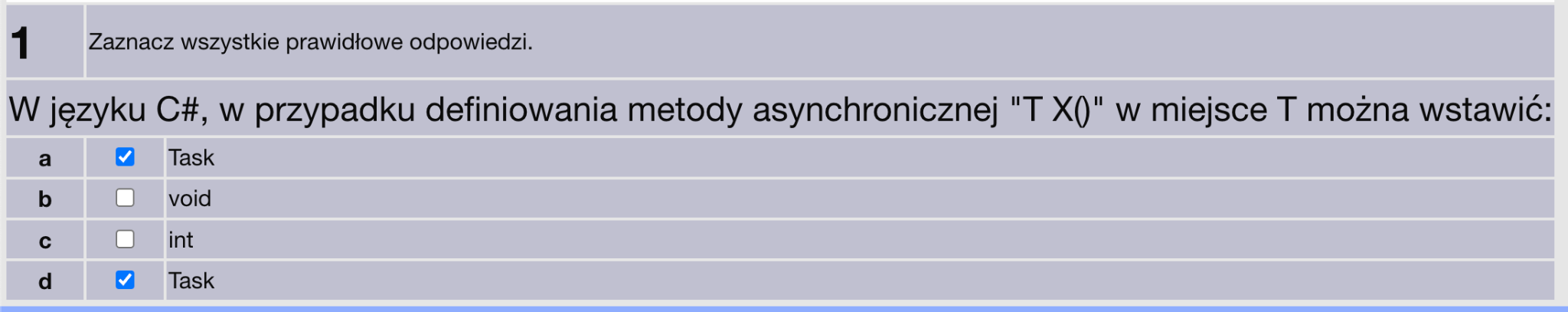
C nie, np. u nas apka lekarza i aptekarza, ta sama platforma web

D nieprawda, mikrousługa aplikacyjna może oddziaływac z apkami na róznych platformach (web / mobile / itd)

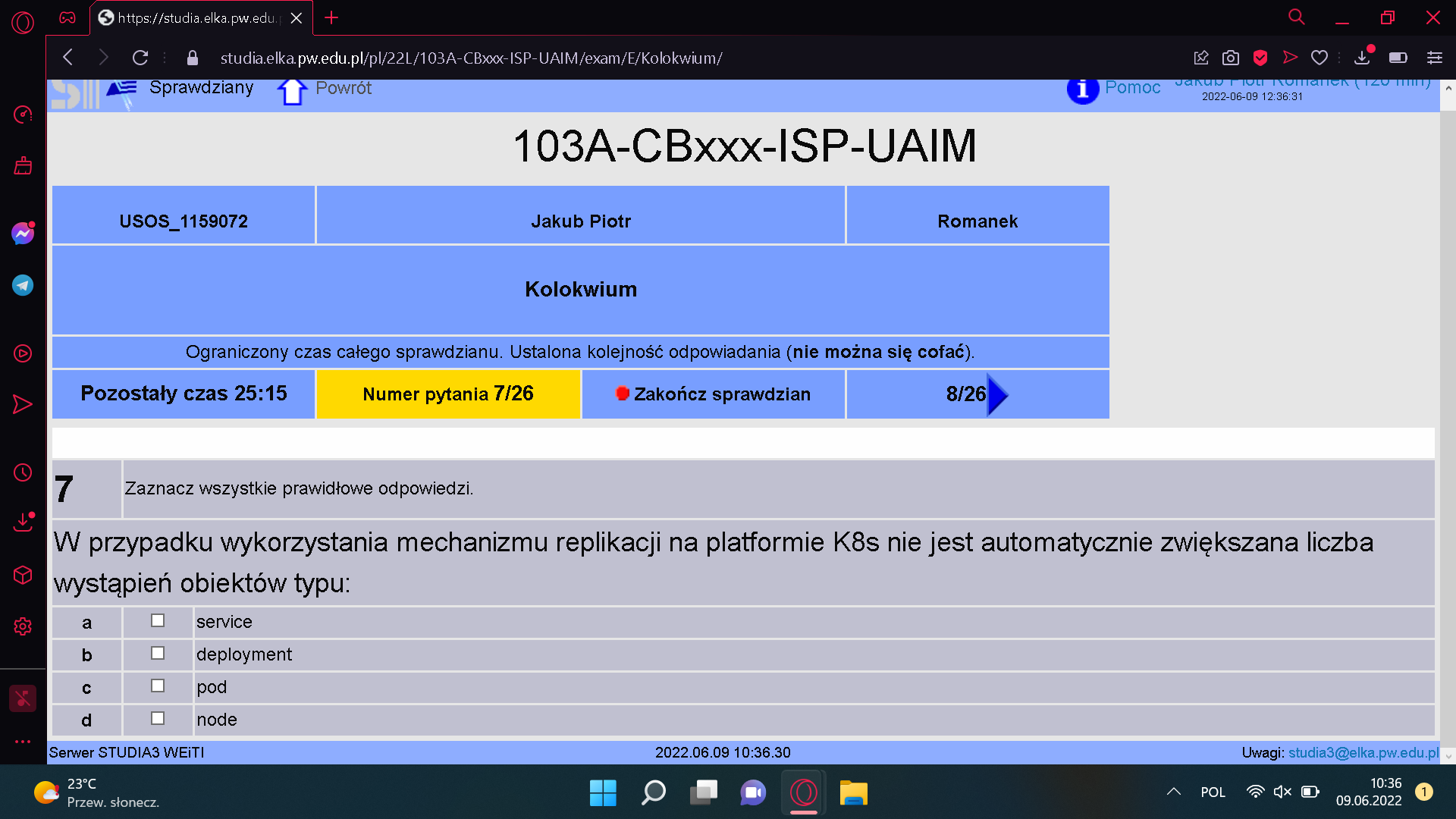
::

Platformy: web/desktop/mobile



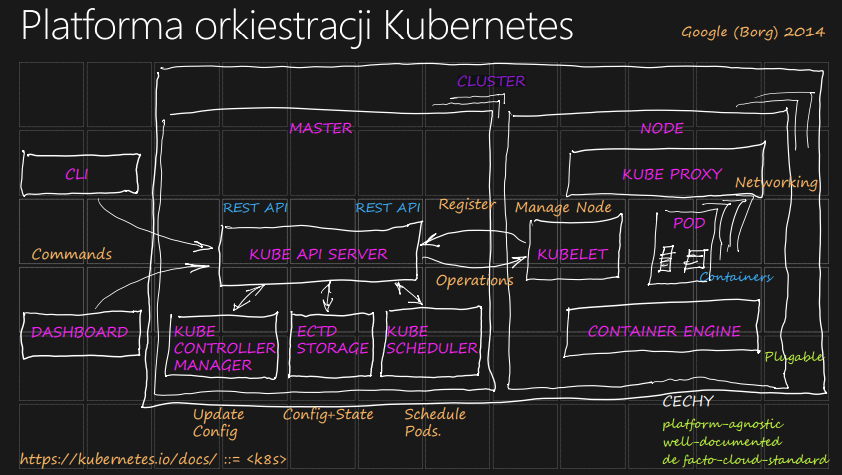


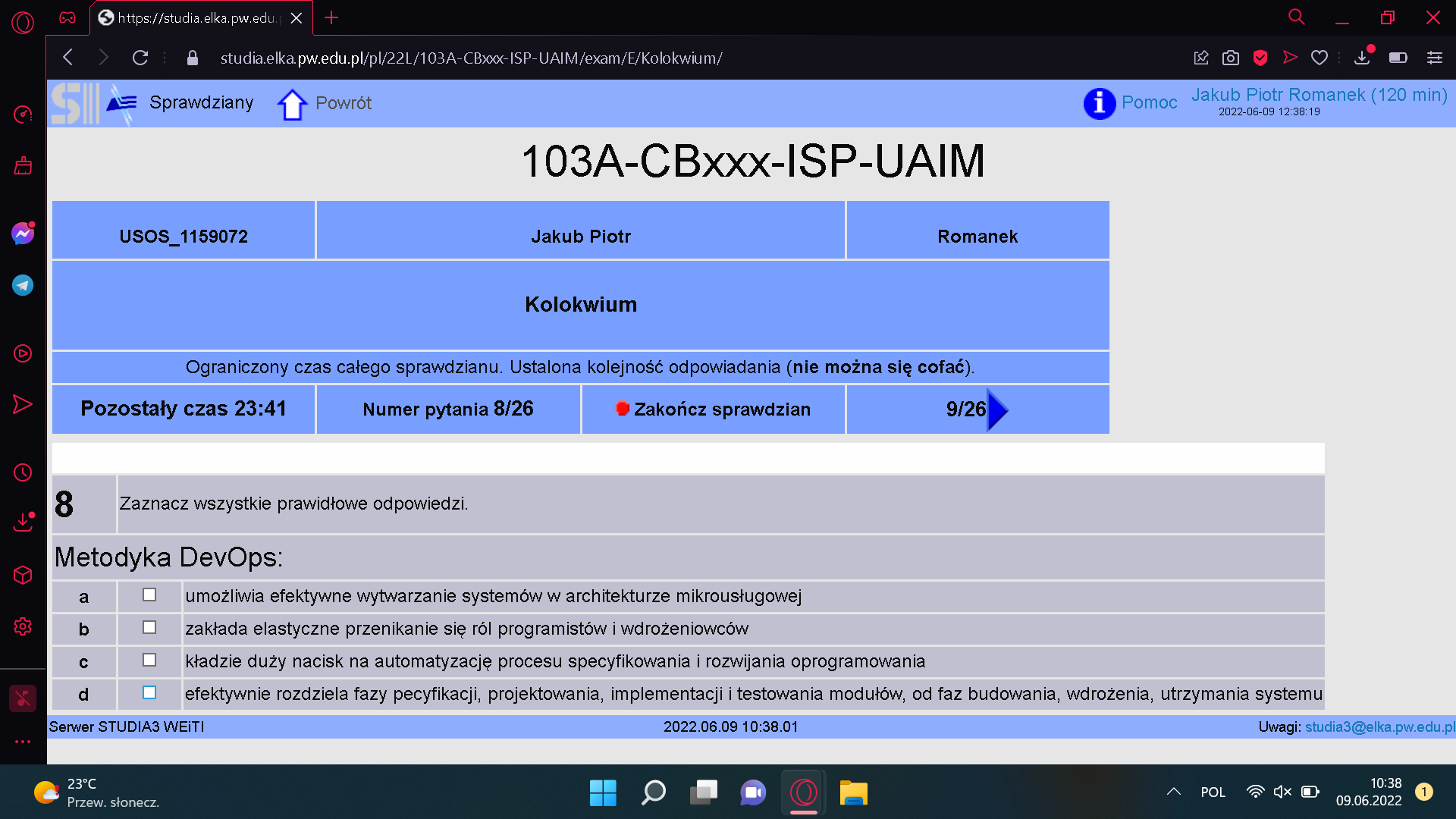
Task oraz void



Replikacja: nie zwiększana liczba: service, deployment, node ? ale raczej ok

//replikacja – pod



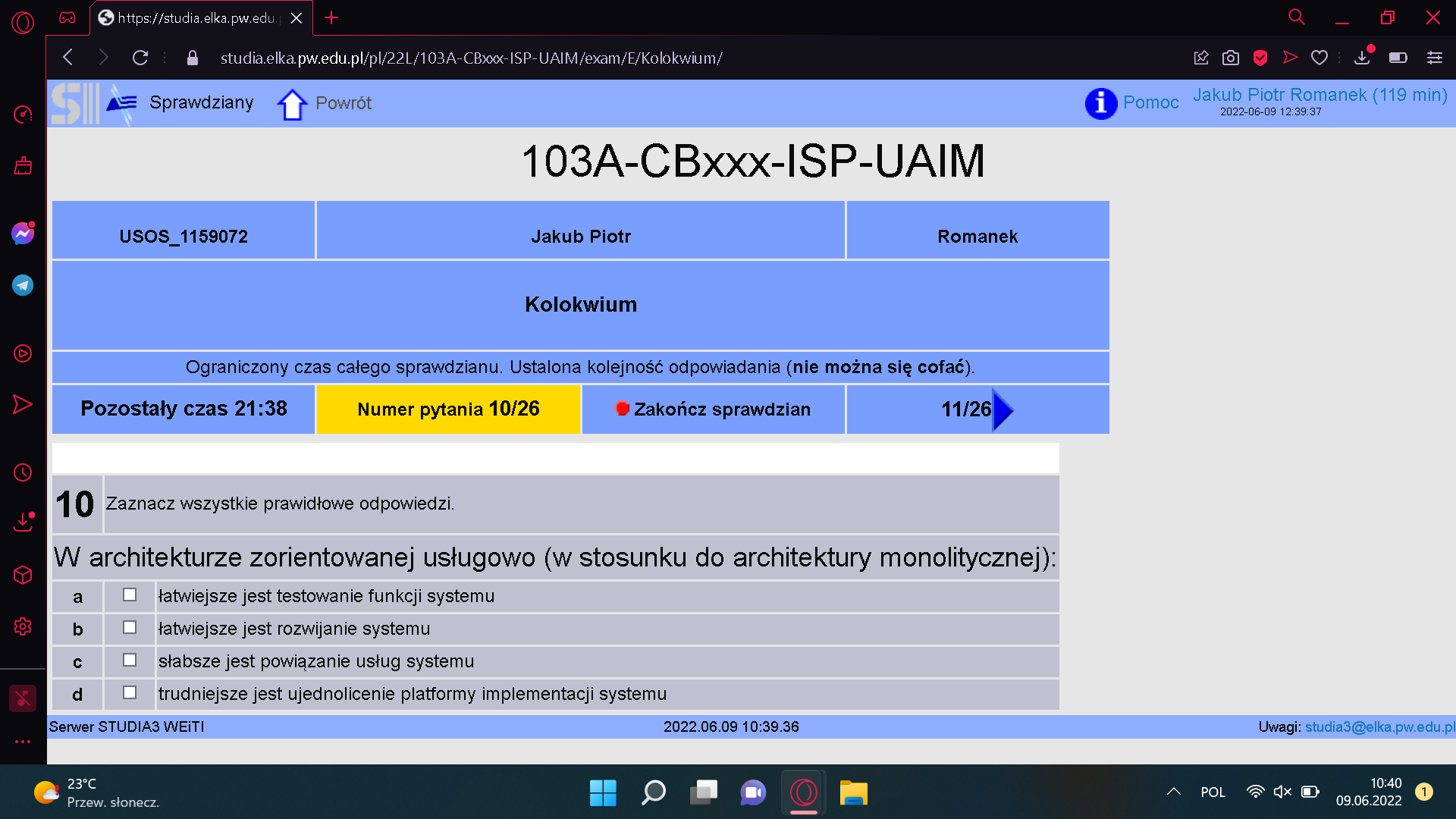


A – tak

B – tak

C – raczej tak

D – nie



A – tak, easier to debug and test than are huge chunks of code like in the monolithic approach

B – tak, rozwój lepszy

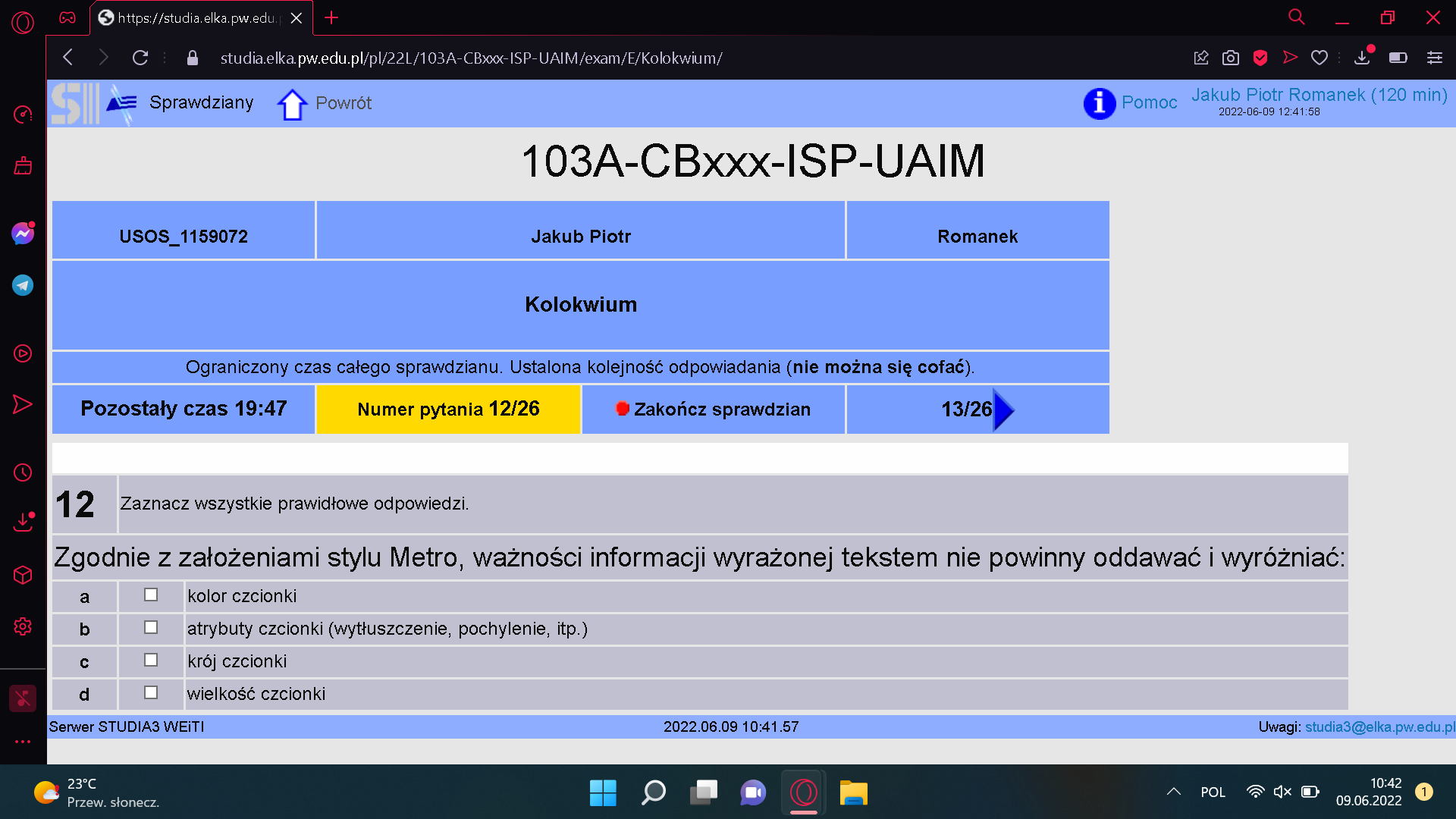
C – tak

D – rzekomo prawda, bo są oddzielne części ?

Obraz zawierający tekst

Opis wygenerowany automatycznie

C – tekstu // i raczej tylko to, jeżeli ikony to simple icons



Ważność informacji:

* Wielkość czcionki
* Atrybuty jak wytłuszczenie??

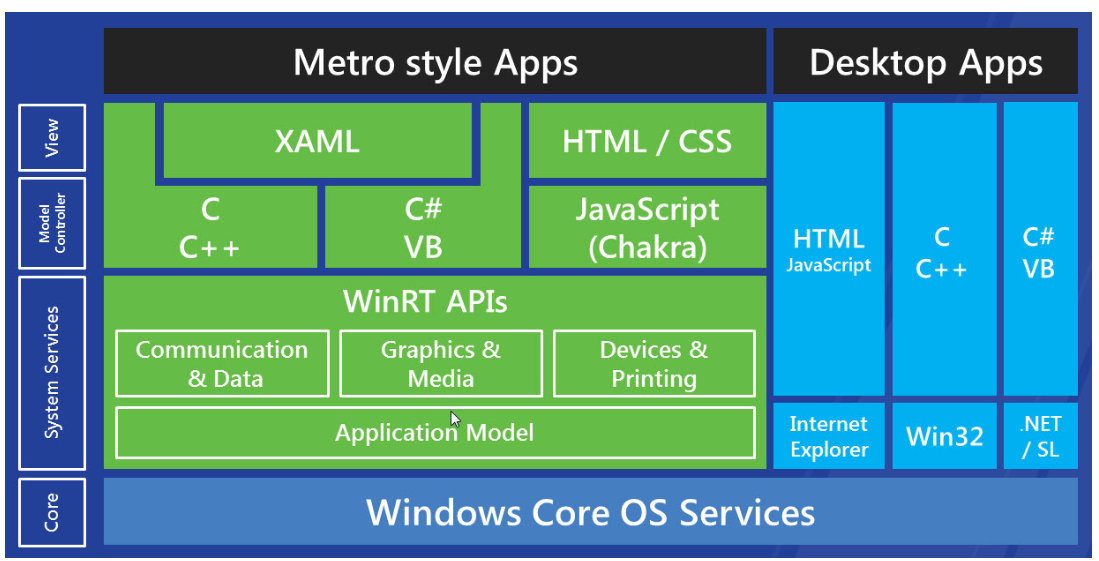
Ważności info nie powinny oddawać:

* Krój czcionki //rodzaj czcionki
* Kolor //to przede wszystkim linki? // a moze jednak? Wazniejsze wtedy

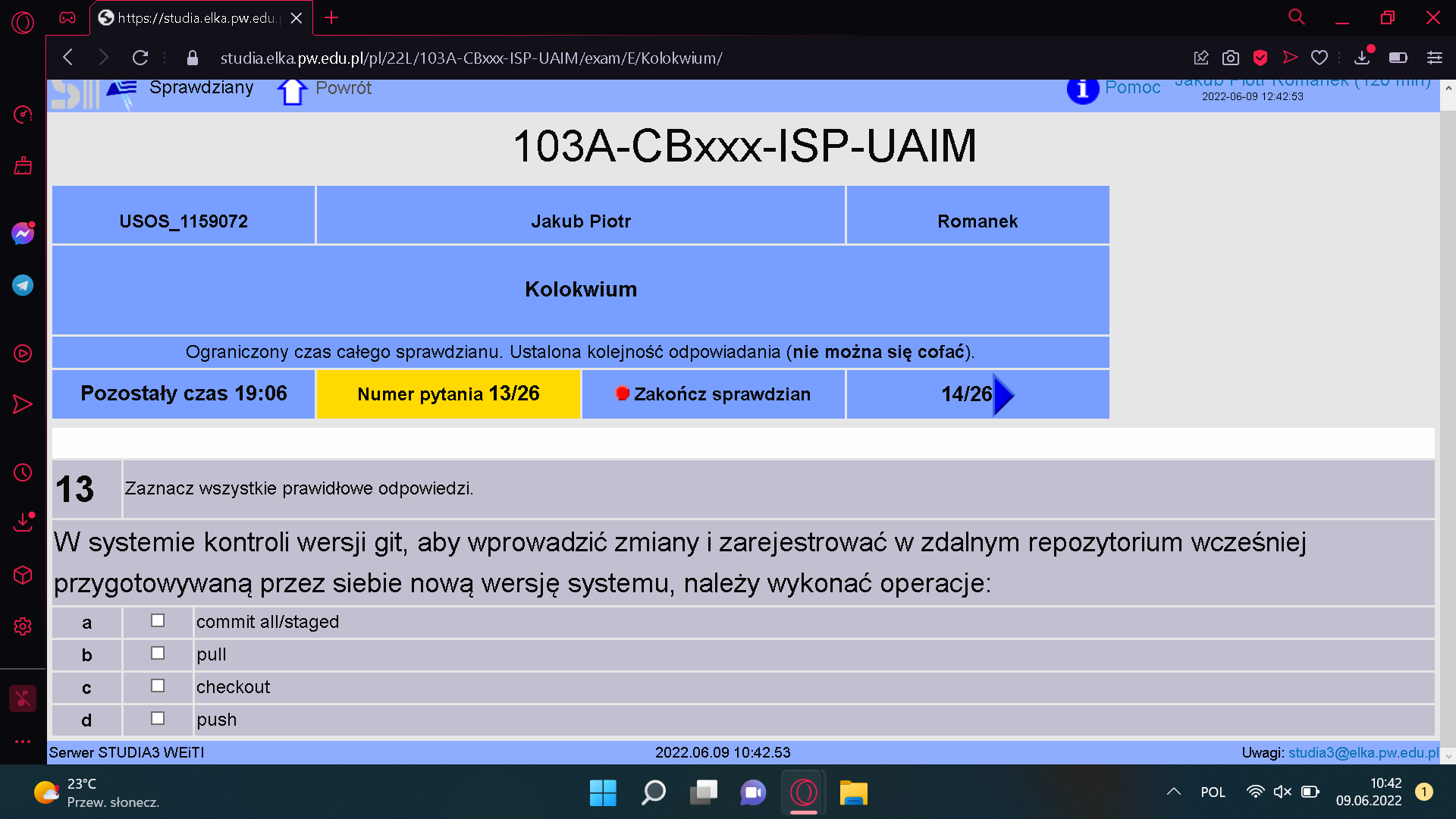
C – nie no, raczej krój nie

D – powinno oddawać ważność info

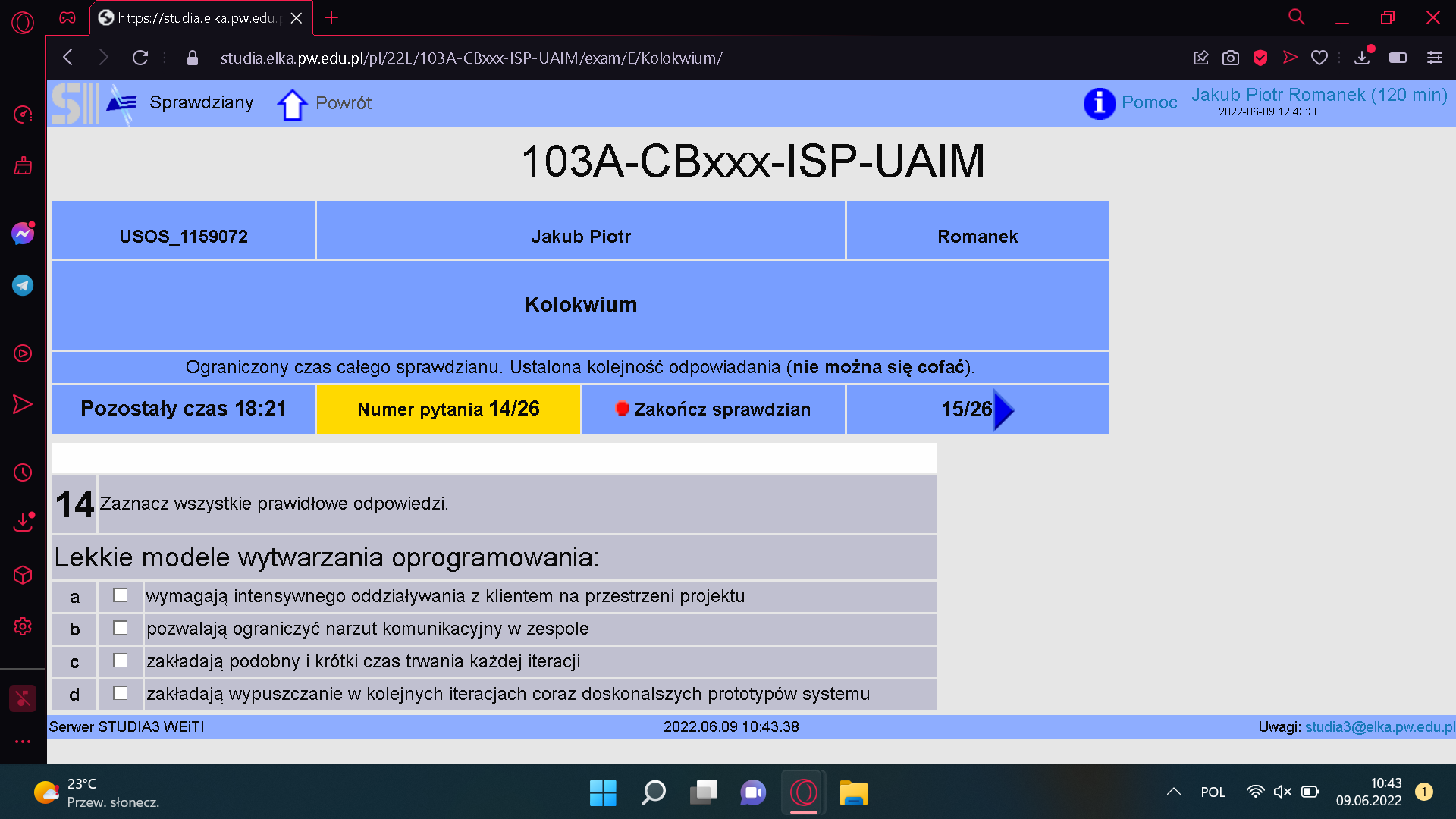
Windows 8 applications are either Metro style Apps or they are Desktop Apps.



Metro apps use the .NET for Metro Style profile and Desktop Apps use the .NET Client profile.



Commit + push

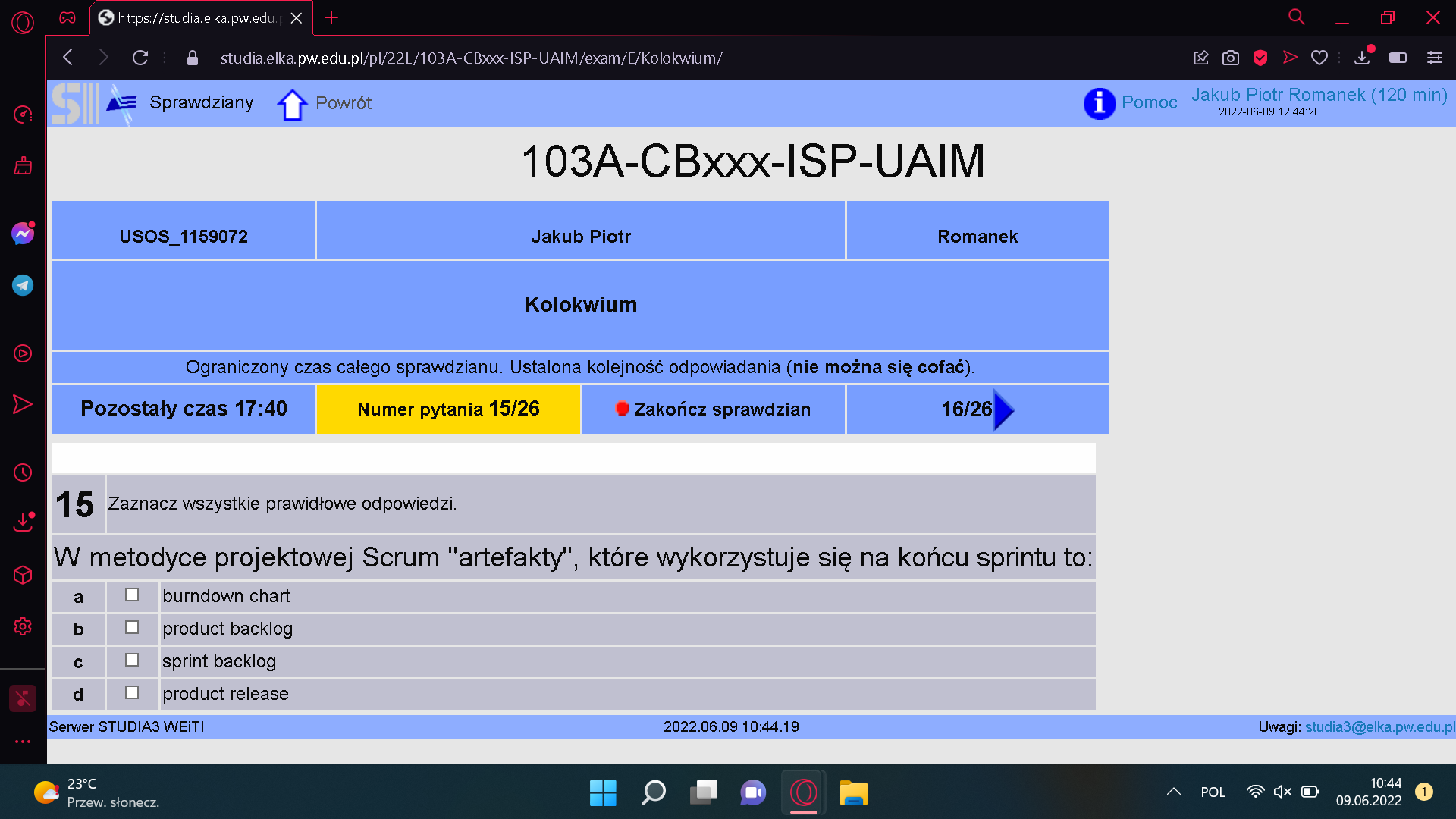


A – raczej na pewno tak

B – nah

C – ok

D – jak najbardziej tak

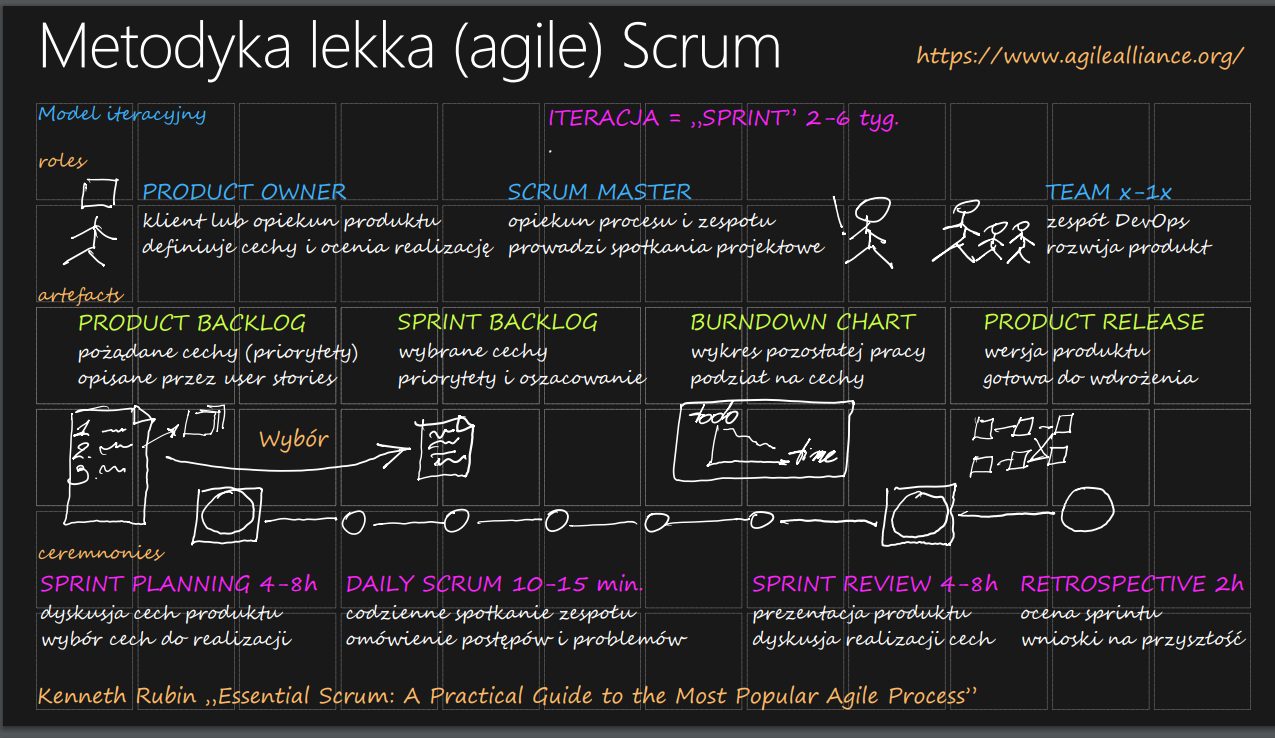


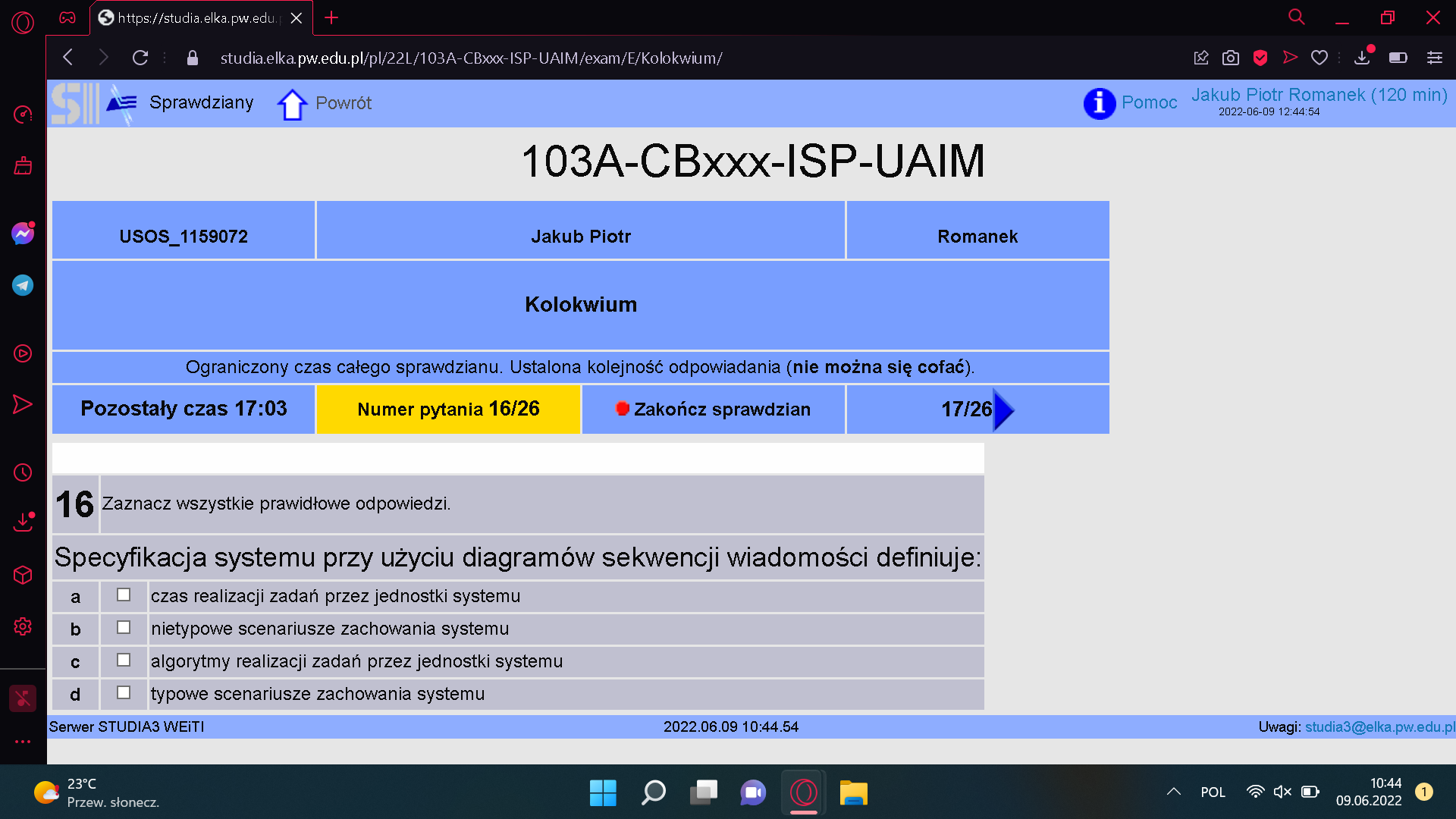
Na końcu sprintu: product release D

Jak zdefiniować koniec?

Product backlog -> sprint backlog -> burndown chart -> product release

Burndown chart – raczej codziennie, analiza tego co zostało



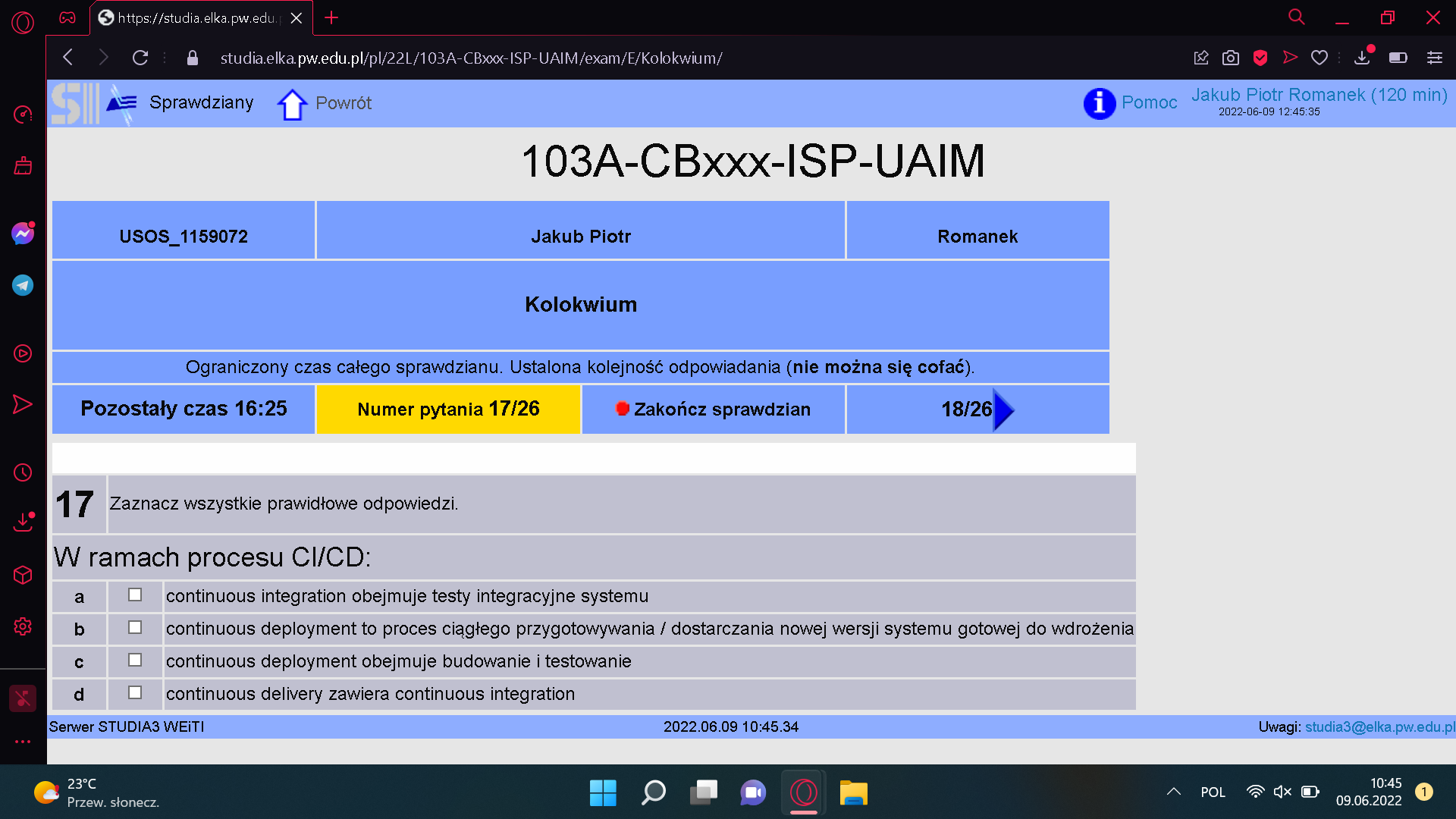


D rzekomo ok

//Definiuje kolejność wiadomości miedzy systemami

//todo: doczytać o diagramie sekwencji

// wsm można zdefiniować timer wiec moze jednak a?



A – wdg red hat - tak

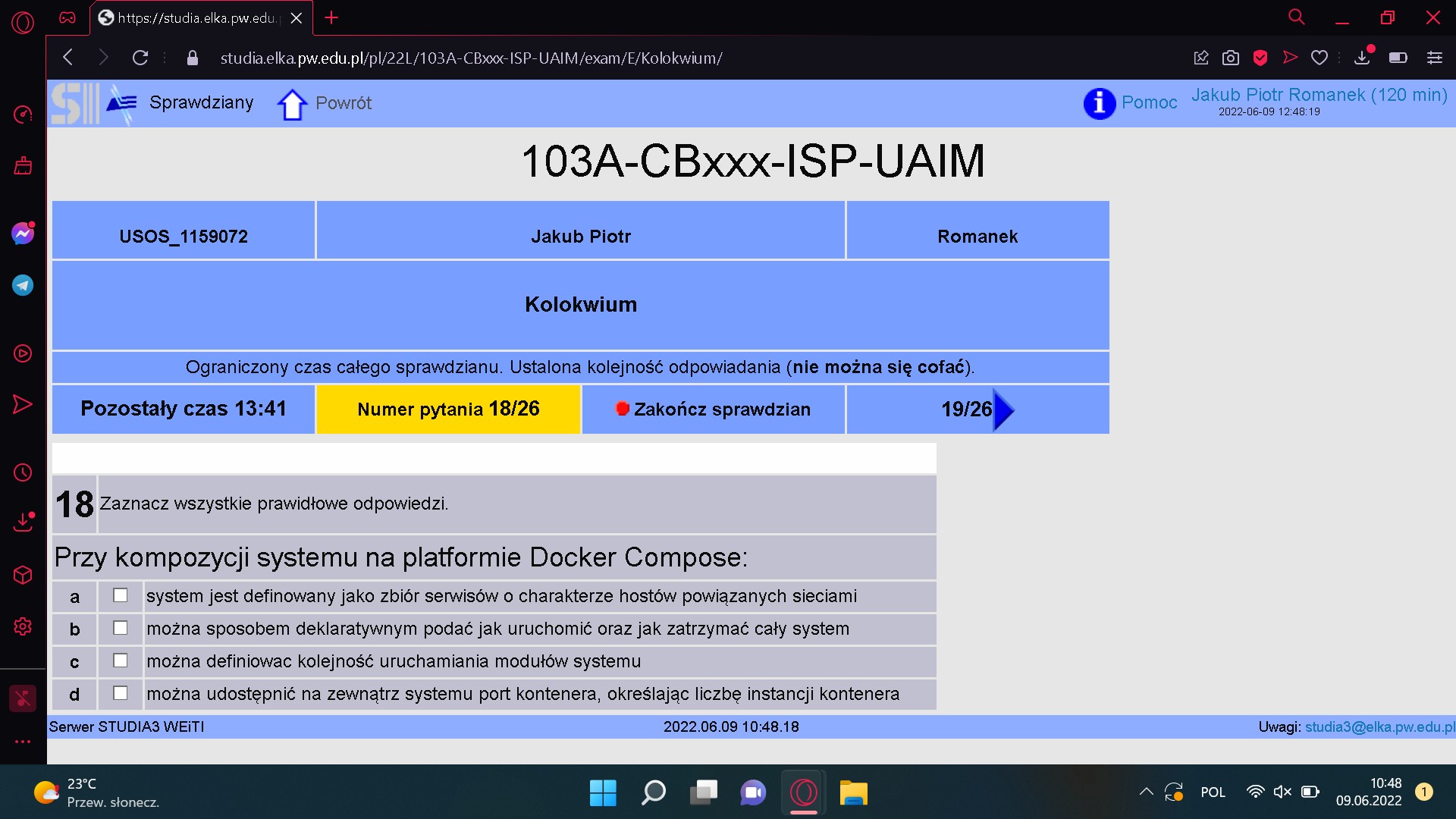
B – to jest delivery

C – cieżko stwierdzić, można uznać, że nie

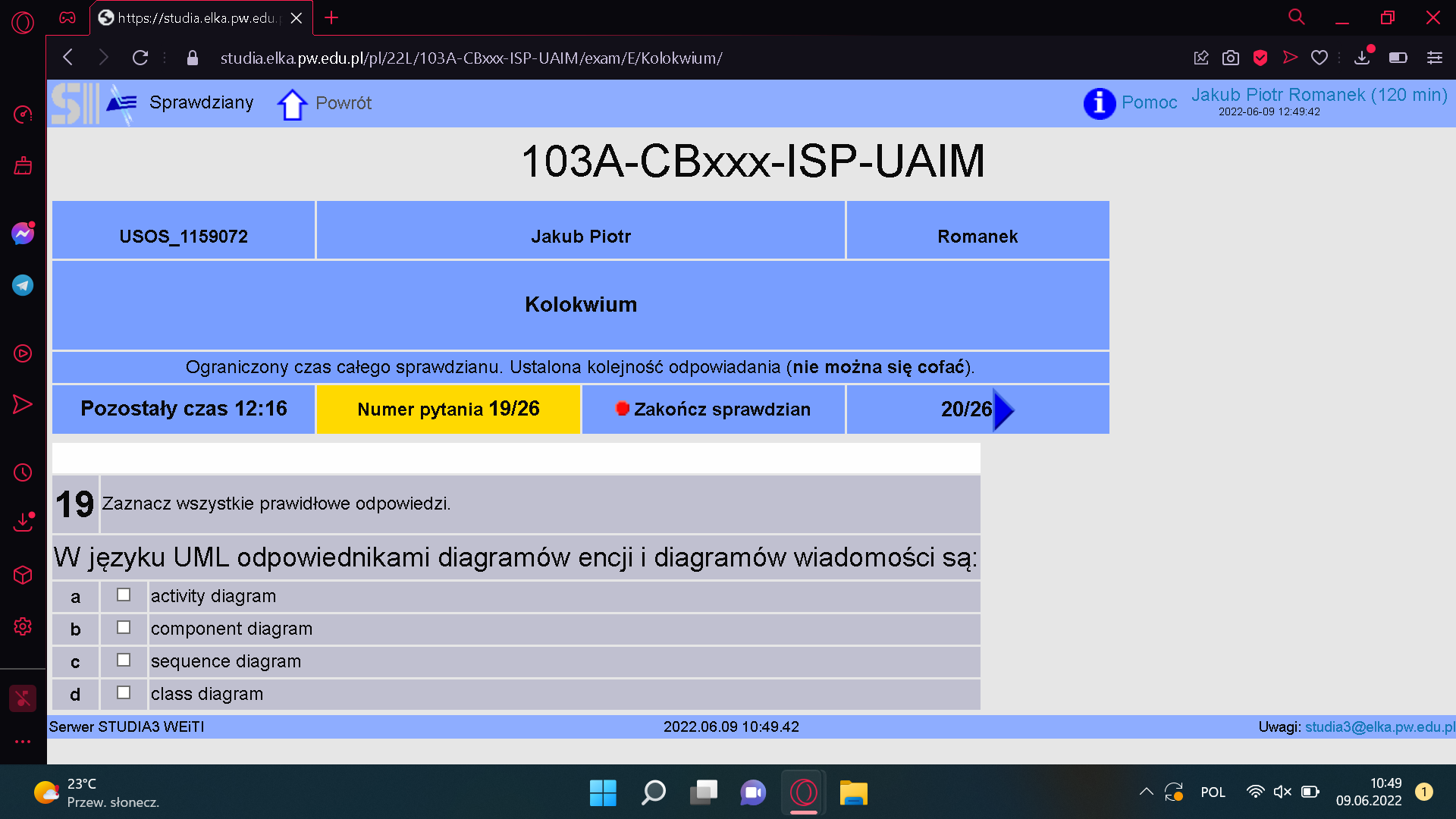
D – cieżko stwierdzić, można uznać, że nie

Unit testing means testing individual modules of an application in isolation (without any interaction with dependencies) to confirm that the code is doing things right. Integration testing means checking if different modules are working fine when combined together as a group.

Continuous Delivery is when your code is always ready to be released but isn't pushed to production unless you make the decision to do so.

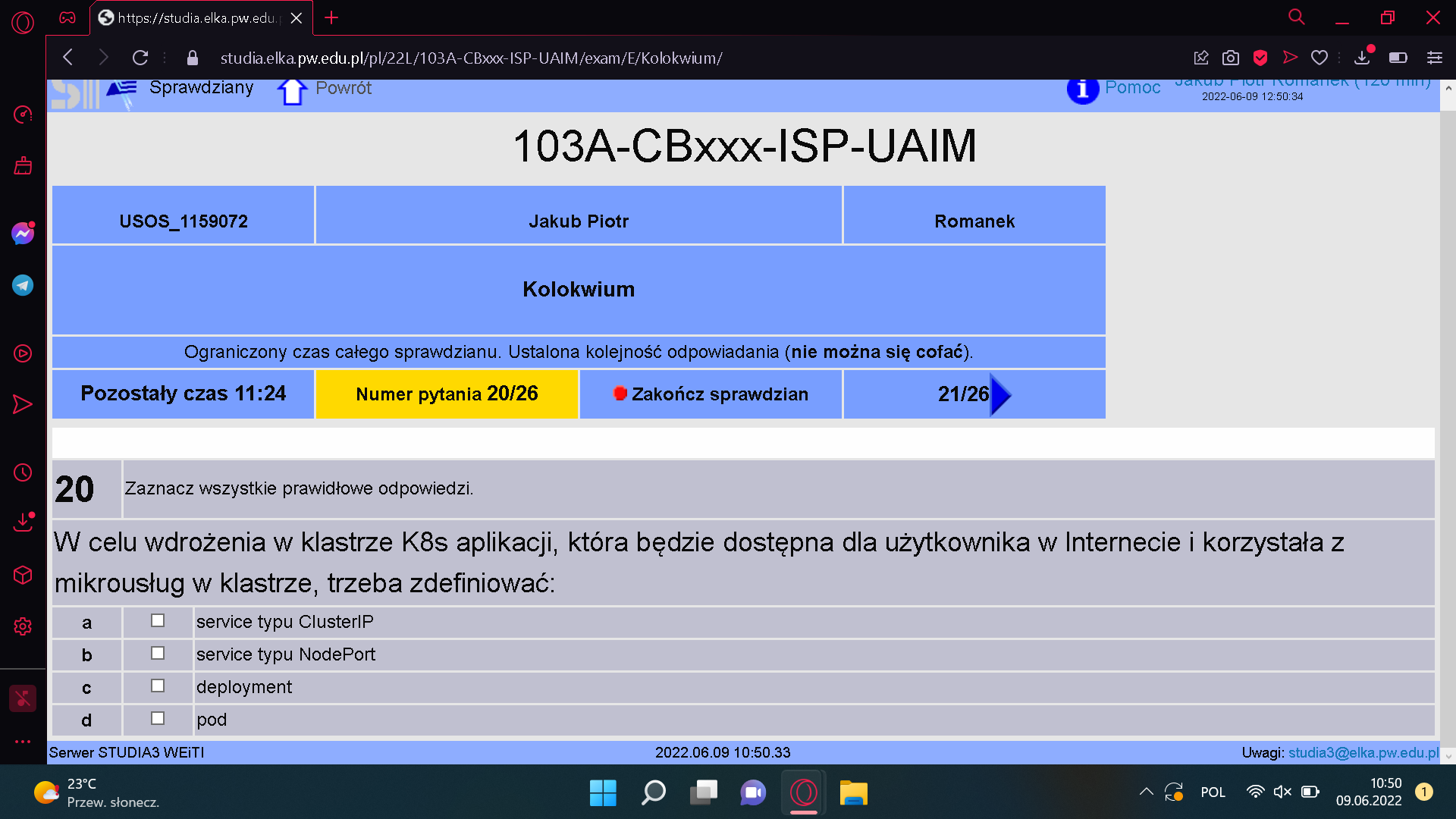


A,b - bezpieczne

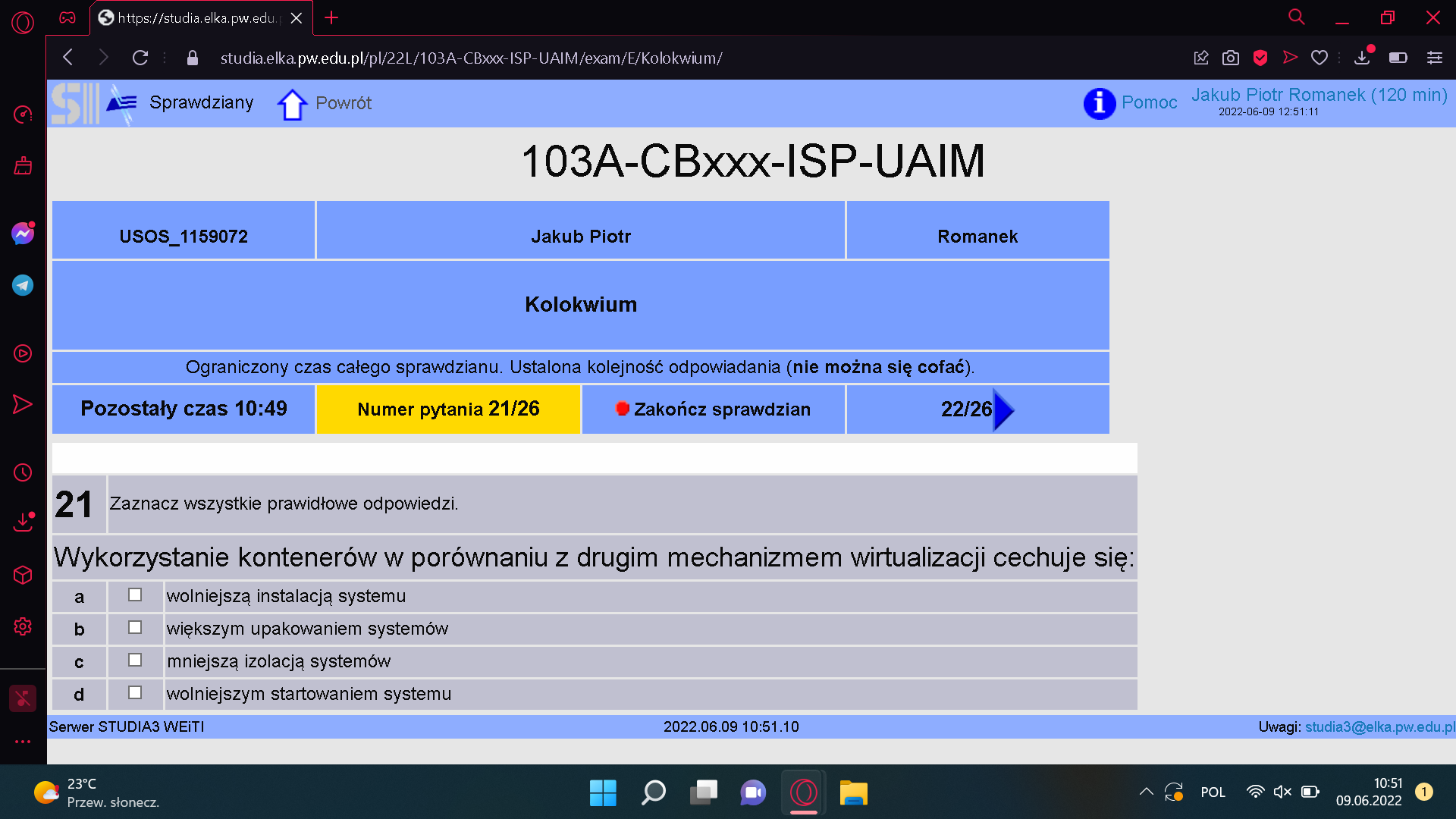


Encji -> klas

Wiadomości -> sekwencji

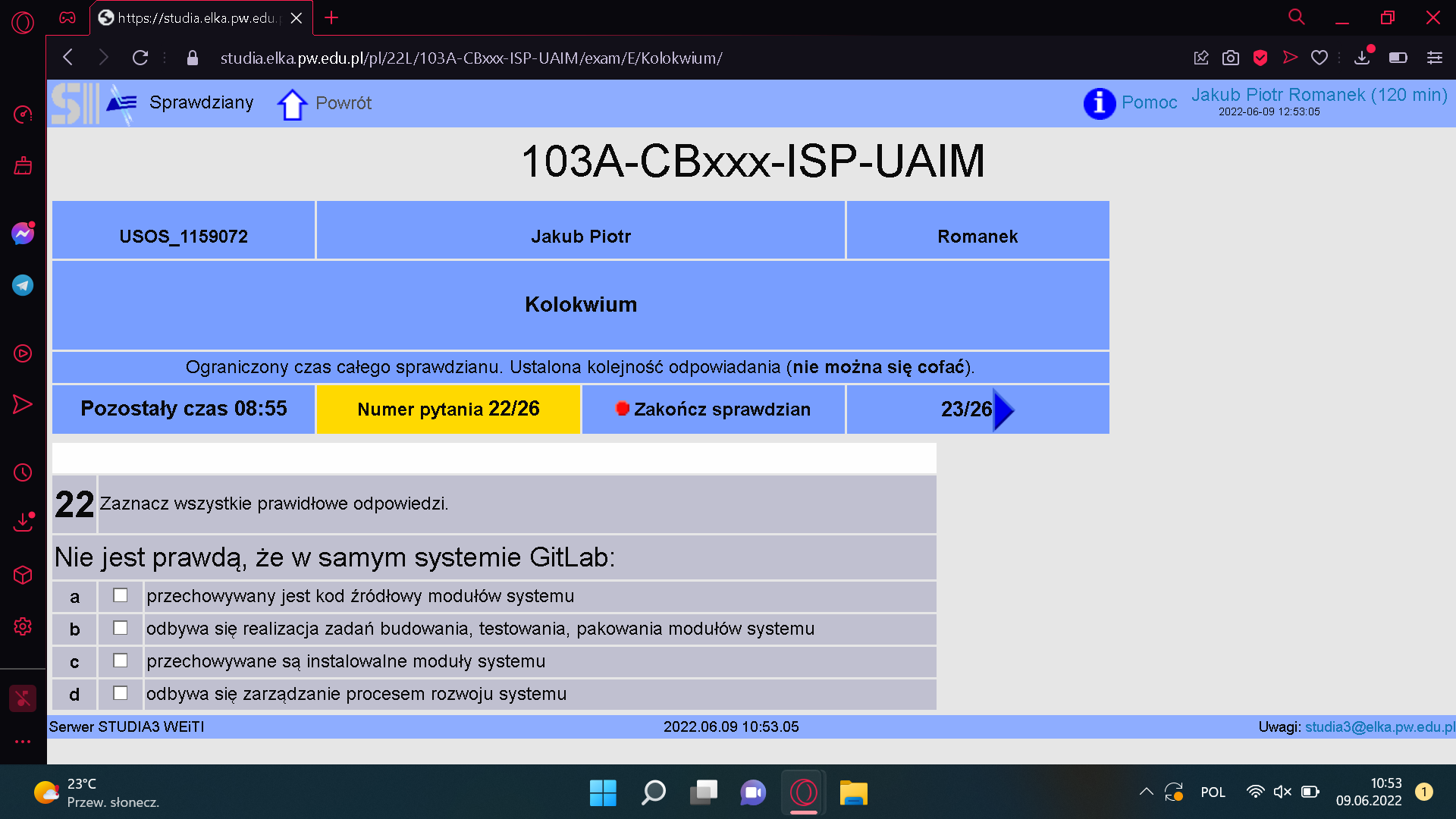


Deployment + nodePort

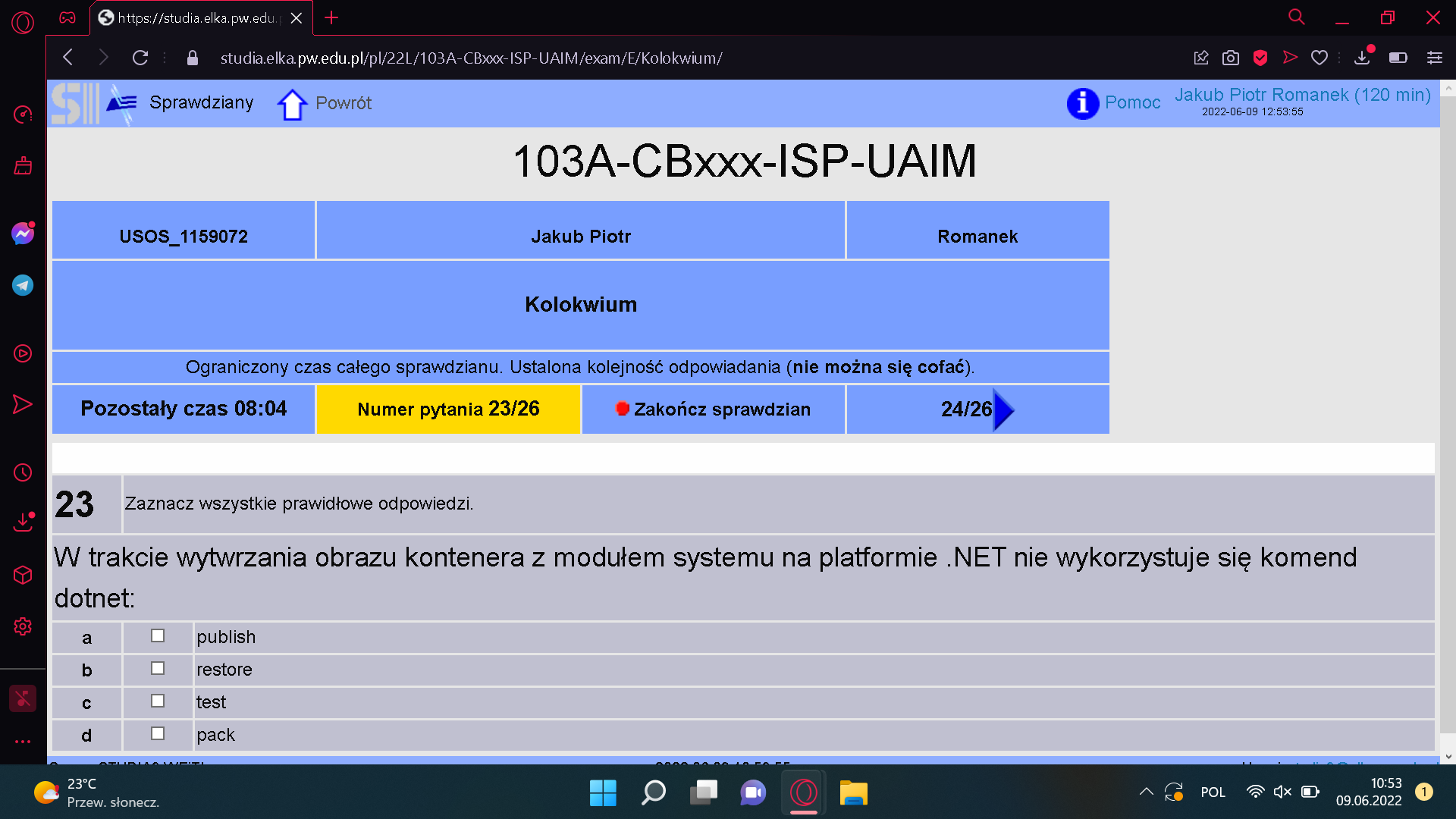


B – większym upakowaniem systemów

C – mniejszą izolacją



C - nieprawda



A Dockerfile is **a text document that contains all the commands a user could call on the command line to assemble an image**. //dotnet commands

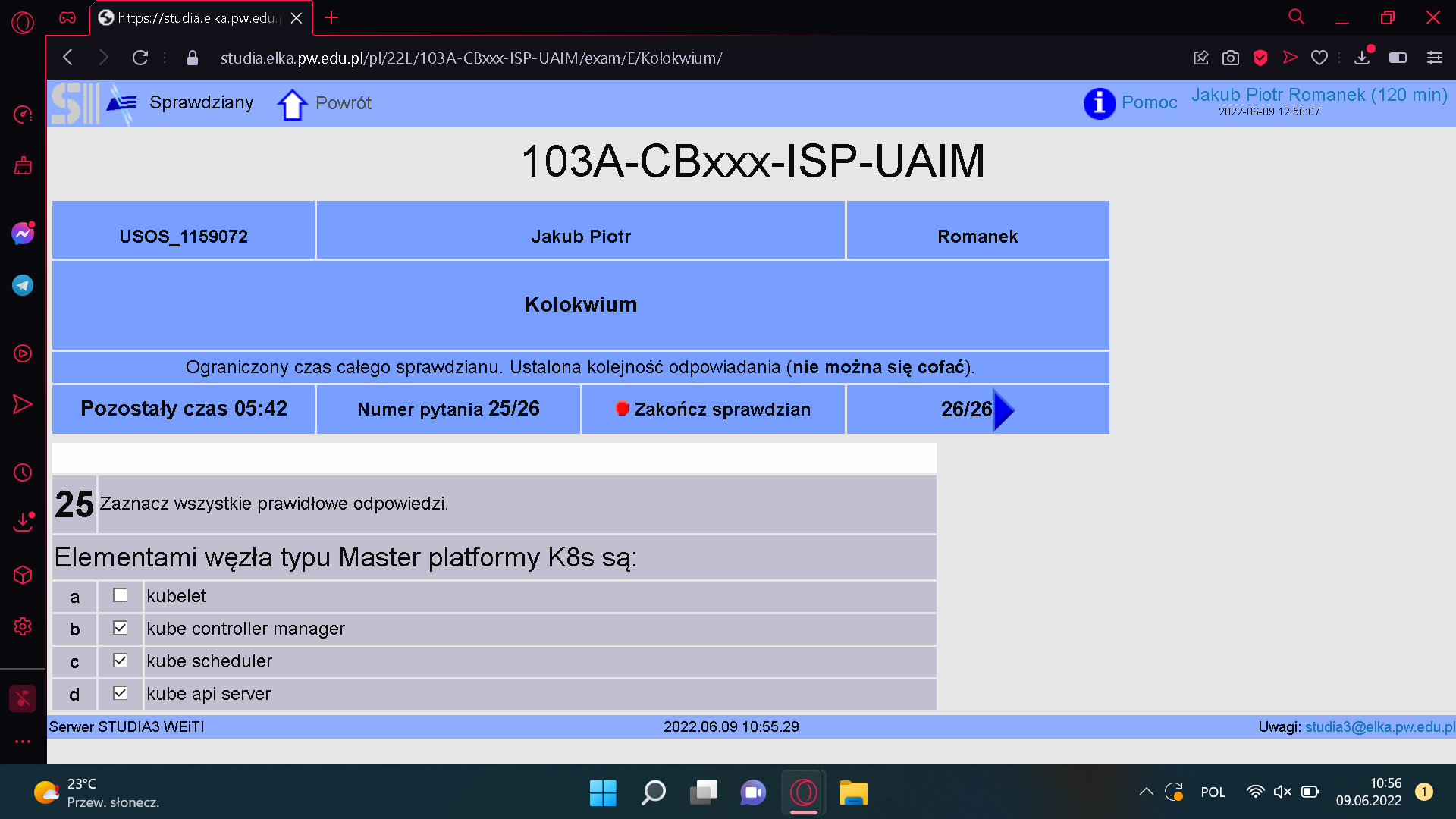
Wytwarzanie obrazu kontenera:

* *dotnet restore, dotnet build, dotnet publish*

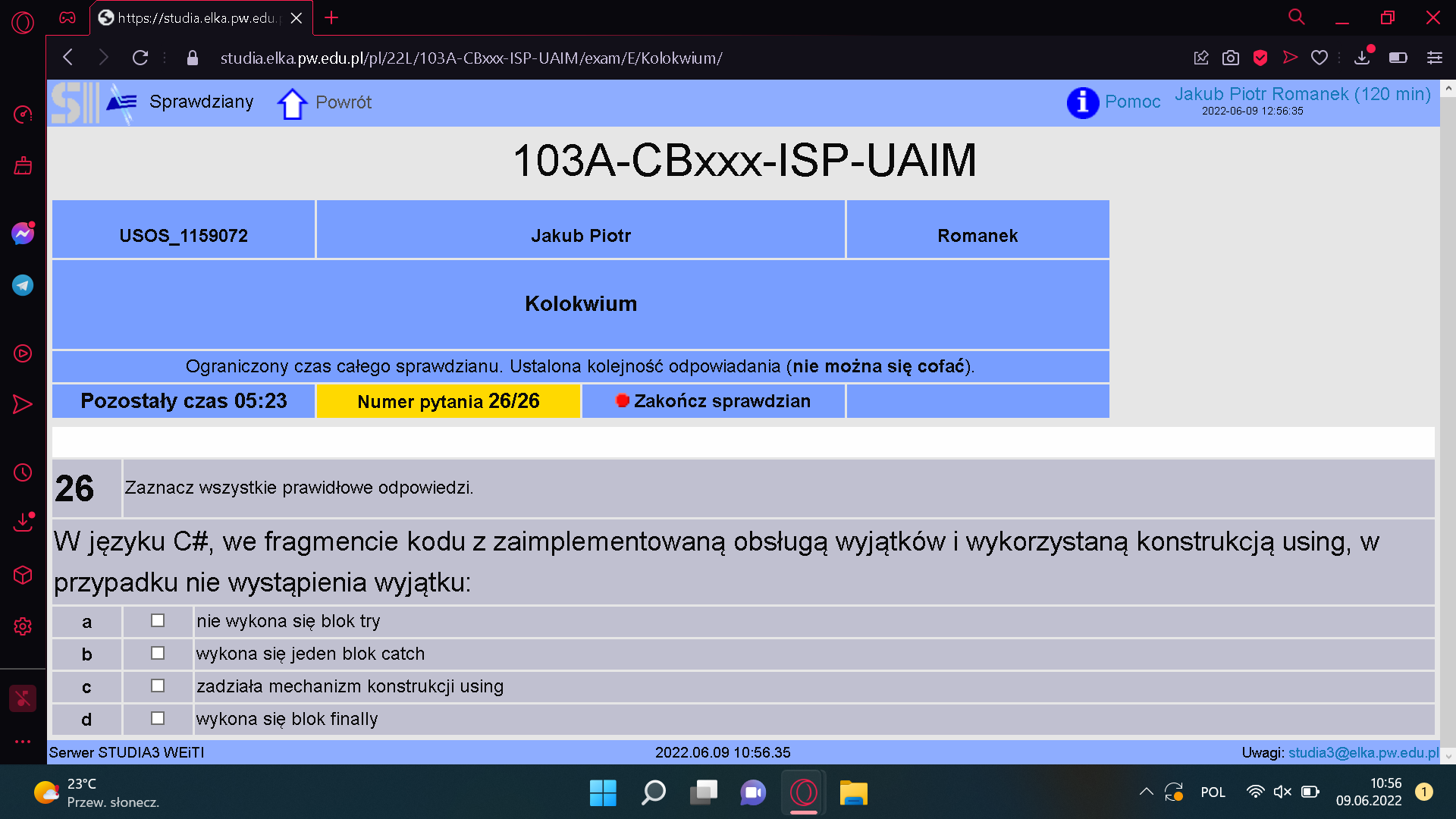
Nie trzeba: pack (dotnet pack builds the project and creates NuGet packages), test (unit tests)

What is difference between *build* and *publish*?

Build compiles the source code into a (hopefully) runnable application. Publish takes the results of the build, along with any needed third-party libraries and puts it somewhere for other people to run it.



Jak zaznaczone

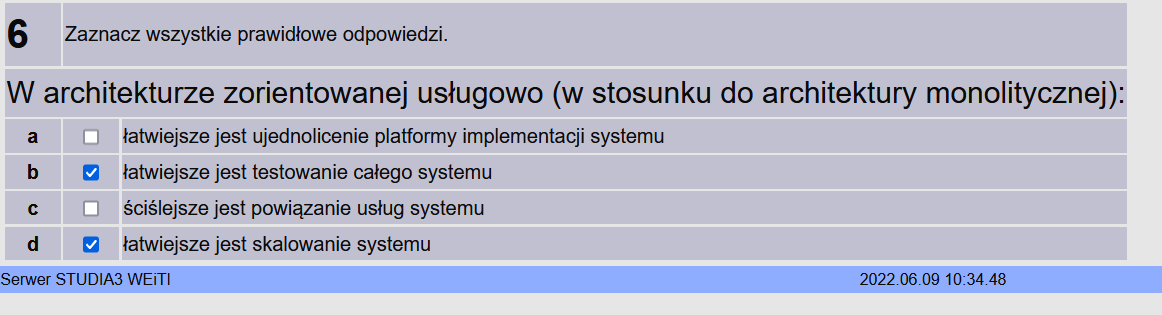


Gdy nie ma wyjątku:

* D – wykona się finally (to zawsze)
* C –zadziała mechanizm konstrukcji using

**The using statement ensures that Dispose is called even if an exception occurs while you are calling methods on the object**.

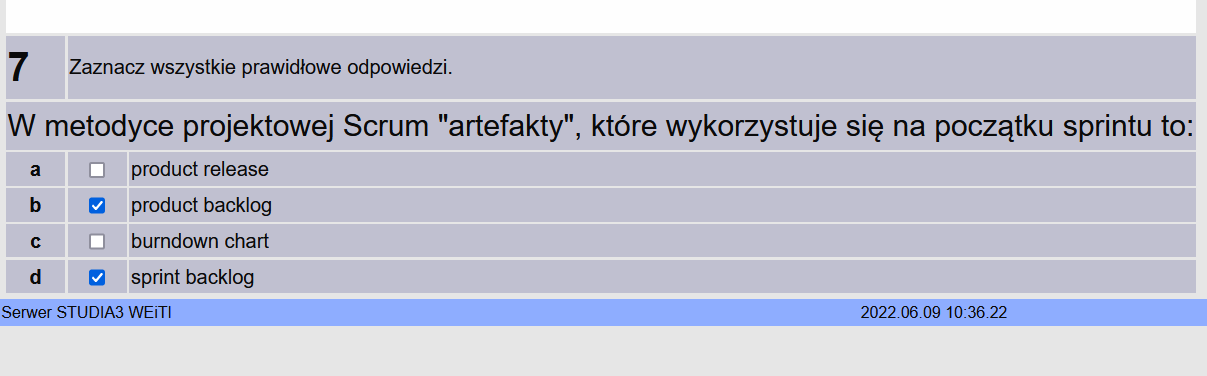
Gdy jest wyjątek, także zadziała mechanizm konstrukcji using.



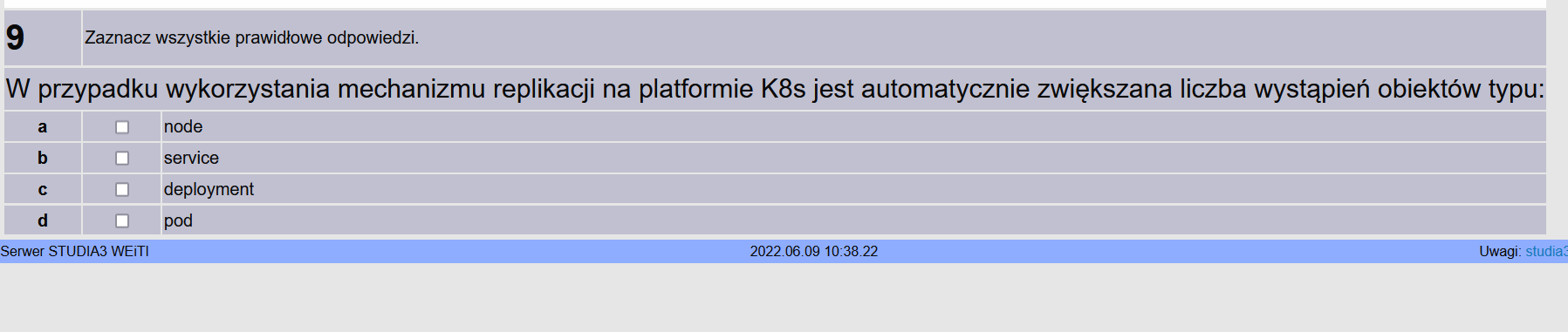
B, D

C na pewno nie

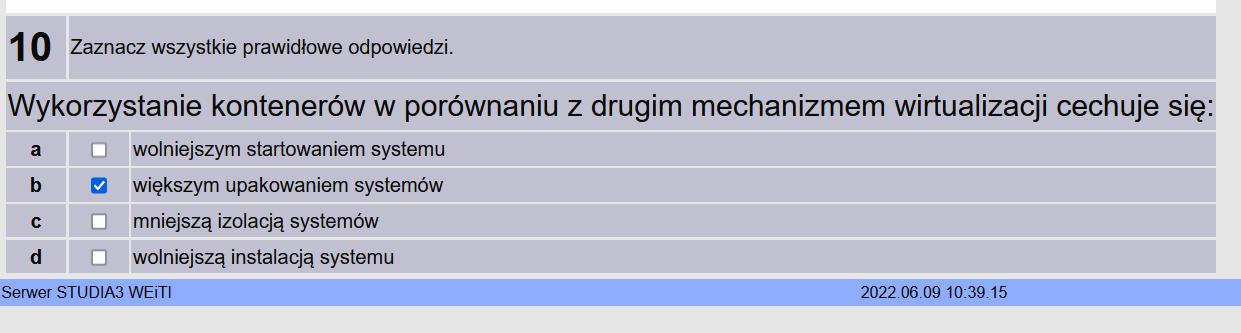
A – trudniejsze ujednolicenie, bo oddzielne mikrouslugi, mogły być w czym innym pisane



Chyba ok



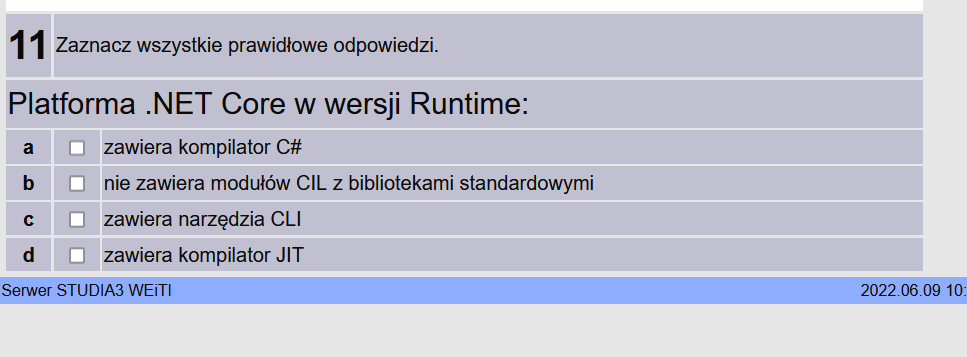
Zwiększane – pod?



Większe upakowanie + mniejsza izolacja?

Obraz zawierający tekst

Opis wygenerowany automatycznie



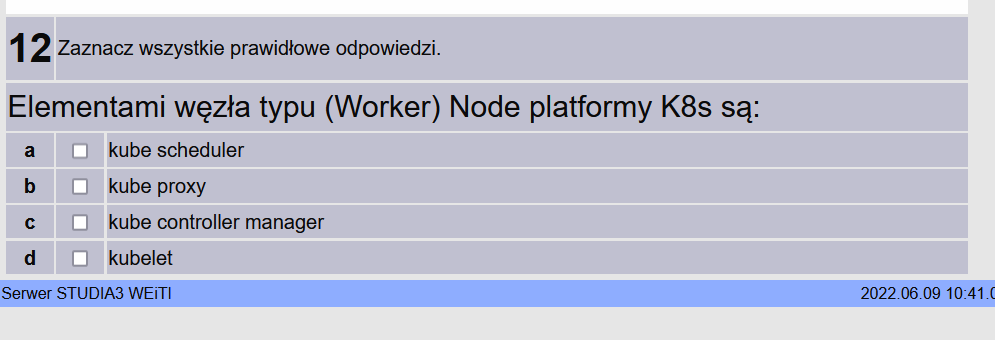
D – zawiera JIT

Raczej CLI też

Raczje moduły też

Kompilator c# nie jest częścią .net framework

Tak więc chyba C,D



Worker node -> kubelet + kube proxy

Ceremonie takie są (agile – metodyki lekkie):

* Sprint planning – wybór cech do realizacji, tutaj oba backlogs
* Daily scrum – codziennie spotkanko, omówienie postępów i problemów, tutaj burndown chart
* Sprint review – prezentacja produktu + dyskusja realizacji cech, product release
* Retrospective – ocena sprintu i wnioski na przyszłość

